Journal of King Saud University – Computer and Information Sciences (2017) 29, 365–377

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

A Variable Service Broker Routing Policy for data center selection in cloud analyst

Ahmad M. Manasrah [a,*], Tariq Smadi [a], Ammar ALmomani [b]

[a] Yarmouk University, Faculty of Information Technology and Computer Science, Jordan
[b] Al-Balqa Applied University, Dept. of Information Technology, Jordan

Received 2 October 2015; revised 12 December 2015; accepted 12 December 2015
Available online 28 March 2016

KEYWORDS

Cloud-Analyst;
Cost effective analysis;
Data centers;
Service broker policy;
Simulation and modeling

Abstract Cloud computing depends on sharing distributed computing resources to handle different services such as servers, storage and applications. The applications and infrastructures are provided as pay per use services through data center to the end user. The data centers are located at different geographic locations. However, these data centers can get overloaded with the increase number of client applications being serviced at the same time and location; this will degrade the overall QoS of the distributed services. Since different user applications may require different configuration and requirements, measuring the user applications performance of various resources is challenging. The service provider cannot make decisions for the right level of resources. Therefore, we propose a Variable Service Broker Routing Policy – VSBRP, which is a heuristic-based technique that aims to achieve minimum response time through considering the communication channel bandwidth, latency and the size of the job. The proposed service broker policy will also reduce the overloading of the data centers by redirecting the user requests to the next data center that yields better response and processing time. The simulation shows promising results in terms of response and processing time compared to other known broker policies from the literature.

1. Introduction and background

Cloud computing is a model for facilitating on-demand network access to shared and configurable computing resources such as servers (IaaS), operating systems (PaaS), applications and services (PaaS) that can be made available and released with less administration efforts or service provider involvements. In a short time, cloud computing has been applied widely in many applications, it became an essential part of the next generation of computing infrastructure at low cost, that enables users to utilize their resources as a pay-per-use as portrayed in Fig. 1.

The main facet of cloud computing is the adoption of virtualization, in which virtual machines (VM) are running on top of the available hardware to satisfy the users need and demand (Kremer, 2013; Armbrust et al., 2010). Therefore, managing VMs is an important aspect to be considered to keep the whole

* Corresponding author.
E-mail addresses: ahmad.a@yu.edu.jo (A.M. Manasrah), tariksmadi@gmail.com (T. Smadi), ammarnav6@gmail.com (A. ALmomani).
Peer review under responsibility of King Saud University.

ELSEVIER Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2015.12.006
1319-1578

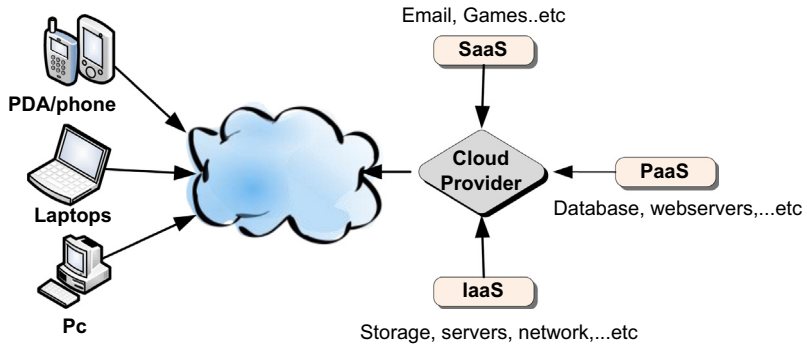

Figure 1 Cloud computing overview.

cloud running efficiently, which is carried out by the hypervisor (Kremer, 2013; Rimal et al., 2009). The selection of the VM for a particular workload is done by the load balancer. The load balancer distributes the load in a way that ensures no VM is swamped with requests at one time (Mell and Grance, 2009; Rajesh and Sreenivasulu, 2014). Above this level, another abstraction level called the service broker, acts as an intermediary between the users and the cloud service providers. The service broker utilizes existing service broker policies to route user requests to the most appropriate data center (Houidi et al., 2011; Limbani and Oza, 2012; Kapgate, 2014; Mishra et al., 2014). Therefore, the optimal response time of a particular request and the efficient utilization of the datacenters are governed through a proper data center selection policy. Especially that, data centers are under the service providers control at diverse locations, which can be configured with different types of hardware based on utilization and clients' demands (Houidi et al., 2011; Limbani and Oza, 2012; Mishra et al., 2014; Sharma, 2014; Mishra and Bhukya, 2014; Rekha and Dakshayini, 2014). The existing broker policies of data center selection are based on the location of the data centers, response time or current execution load, and the cost of the data center usage (Rimal et al., 2009; Dinh et al., 2013). Therefore, the objective of this paper is to illustrate an enhanced proximity service broker policy that selects a data center based on the network latency and bandwidth to ensure efficient and reliable request execution over data centers (i.e. minimized response and execution time).

Cloud computing is a very complex process; it depends on uncontrollable factors like network congestion and servers varying workloads. However, measuring the performance of internet based applications using real cloud platform is difficult (Armbrust et al., 2010; Iosup et al., 2011; Dillon et al., 2010). Therefore, simulation-based approaches are provided to solve such issue virtually and free of charge under stable and controllable environment (Dinh et al., 2013; Wickremasinghe et al., 2010). Calheiros et al. (2011), proposes an extensible toolkit for modeling and simulating cloud computing systems called CloudSim. The CloudSim provides a set of components that provide the base for cloud computing, including Virtual Machines (VM), Cloudlets (Jobs and user request will be used interchangeably), datacenters (DC), Service broker and hosts. Each of them has its own characteristics and functionality. DCs consist of a number of physical hosts, each of which manages a number of allocated VMs. There is a policy to maintain the efficiency of the VM allocation. CloudSim offers a straight forward policy, which is a first-come-first-serve (FCFS) policy. The Datacenter within the CloudSim has its own characteristics like architecture, OS, list of Virtual Machines, allocation policy (time-shared or space-shared), time zone and the cost of the provided services. Virtual machine parameters are: Size, Ram, MIPS (Million Instructions per Second) and bandwidth. Cloudlet (job) parameters are: length (Number of Instructions), user, input file size and output file size (Calheiros et al., 2011). Users within the user base generate their own requests and send them as a job to the cloud through the Service Broker. The service broker selects an appropriate data center according to the service broker policy. The result is returned back to the user through the service broker in a reversed order after the job is finished (Limbani and Oza, 2012; Wickremasinghe et al., 2010). Therefore, this paper proposes a service broker policy for datacenter selection with the best possible response time, delay and bandwidth (i.e. availability) to serve the user's requests. The algorithm balances between the delay, bandwidth and the request size in selecting the most suitable data center. Few scenarios were conducted to introduce heavy and light loads on the datacenters. The proposed algorithm shows an enhancement in response and processing time compared to other algorithms but almost similar in terms of the overall cost. In this paper we present some enhancement over the service proximity broker policy through taking the communication channel bandwidth, latency and the size of the job into account in an attempt to come up with a new service broker policy that achieves minimum response and processing time.

The rest of the paper is organized as follows: Section 2 discusses the existing service broker policies and the datacenter selection algorithms, hence, the problem formulation. Section 3 demonstrates the proposed service broker policy. Section 4 discusses the simulation environment setup and description as well as discussing the results of the simulation using the new proposed policy. Section 5 concludes the topic and Section 6 provides an idea on the future scope.

2. Data center selection problem

Since the main goal of the service brokers is to direct the user requests to the best DC with optimal performance, the service broker policy has to efficiently select the best data center for the job considering many factors such as time, cost, and

availability. Cloud analyst is an open source toolkit built directly on top of CloudSim toolkit (Wickremasinghe et al., 2010; Calheiros et al., 2011). Cloud analyst provides the researchers with various tools and methods to simulate the cloud and evaluate its service performance (Limbani and Oza, 2012; Calheiros et al., 2011). For the purpose of directing the user request to the best DC, Cloud analyst includes three routing policies: (1) network latency-based "*i.e. service proximity based*" routing (Limbani and Oza, 2012; Sharma, 2014), (2) response time-based "*i.e. Performance Optimized*" routing (Zhang et al., 2010) and (3) Dynamic load-based "*i.e. Dynamically reconfigure*" routing (Rekha and Dakshayini, 2014). In Service Proximity Based Routing, the broker chooses the shortest path from the user base (UB) to the data center (DC) based on the network latency only. This may result in overloading the closest data center and its communication channel because it does not consider the channel bandwidth. While in performance optimized routing policy, the broker chooses the best path based on network latency and DC workloads, to achieve the best response time based on the last job response time. This will be generalized as the status of any other DC. However, if any DC with a current load of zero, it will not be selected unless a certain amount of time is waited (i.e. Cool-Off-Time). This could leave the DC idle with no jobs assigned even if it was on the closest (i.e. least latency) and highest available bandwidth network path (Rani et al., 2015). On the other hand, the dynamically reconfigure routing is similar to the proximity based routing, but the broker scales the application deployment based on the load it is facing (Limbani and Oza, 2012; Wickremasinghe et al., 2010; Semwal and Rawat, 2014).

From the above, we concluded some shared drawbacks between the three service broker policies as summarized in Table 1. For instance, in service proximity based routing, the service broker doesn't take into account the request data size or the network bandwidth, which could degrade the overall performance significantly especially in the case of big data or multiple requests that share the same communication channel and bandwidth (i.e. file or gaming servers) (Rekha and Dakshayini, 2014). On the other hand, performance optimized routing has the same issues as the service proximity routing policy. It considers the servers load based on a previously performed jobs and selecting them accordingly, regardless of the network bandwidth and the job size (Sharma, 2014; Ahmed, 2012). Finally, the dynamically reconfigure routing is not efficient if the number of regions and the number of data centers are limited; because it scales the applications deployment based on the current load (Rekha and Dakshayini, 2014). As a result, several researches were conducted to enhance the brokerage policies in terms of processing time, response time, workload, cost, bandwidth, etc. Therefore, Limbani and Oza (2012), proposed a service broker policy that aims to select the data center of the lowest cost within the same region of the user base. This policy is efficient in selecting the lowest cost data center. However, it is still incompetent because it has no consideration for the response time, file size, bandwidth and the work load (Limbani and Oza, 2012; Rekha and Dakshayini, 2014). Therefore, Semwal and Rawat (2014), proposed a new policy to select the data center with the highest configuration. The main goal of this policy is to optimize the response time. However, this goal was achieved but at the same time increases the overall cost if the data centers process huge data (Kapgate, 2014; Mishra et al., 2014; Rekha and Dakshayini, 2014; Semwal and Rawat, 2014).

Kapgate (2014), proposed a DC selection algorithm based on the number of times the data center is selected according to its processing capacity, instruction and memory requirement of the upcoming requests. The author mainly focuses on reducing the associated overhead, service response time and improving overall performance. Even though the proposed algorithm improves the performance of the existing proximity algorithm, it still increases the overall cost (Kapgate, 2014). To resolve the previous issue, Sharma (2014), applied the Round-Robin load balancing policy to distribute the workload among multiple available data centers within the same region. The Round-Robin load balancing policies strive to equalize the total cost of the data centers. Vaishali Sharma results show efficient resource utilization under the proposed simulated environment. However, this may not always be the case if the data centers have different configurations (Sharma, 2014). Consequently, Mishra et al. (2014), proposed an extended Round-Robin service broker algorithm that aims to distribute the requests based on the rating of the data centers, to enhance the overall cost with minimal response time. The proposed method is better than the random selection algorithms. However, if there are some data centers faster than others, they will be selected quite often, and hence get overloaded (Mishra et al., 2014; Sharma et al., 2013).

3. Proposed service broker policy

Generally, Cloud environments are populated with a huge number of heterogeneous data centers that communicate with each other in an ad-hoc manner to provide the intended services to the end users. On the other hand, the user's satisfaction is measured by the QoS of the provided services. Therefore, the availability of data centers and the reliability of the services are important for better quality of services (QoS). Unfortunately, data centers might be overloaded (i.e. resource shortages) due to inequitable data center selection, load distribution or the increased number of user's and their requests. The influence of the overloaded data centers can be

Table 1 Highlights and drawbacks of previous and proposed service broker policies.

Policy name	Factors considered in each policy				
	Available bandwidth	Latency	Job size	Execution time	DC current load
Service proximity	No	Yes	No	No	No
Performance optimized	Yes	Yes	No	No	Yes
Dynamically reconfigure	No	Yes	No	No	Yes
Proposed	Yes	Yes	Yes	Yes	Yes

observed through a degraded QoS (Beloglazov and Buyya, 2013). As a result, overloaded servers will drop new incoming requests (i.e. buffers are saturated) and new connections are refused (i.e. Queues are full). Since the response time is an estimation of the needed time from the moment a user sends a request to a data center, to the moment the user starts receiving the results, a high response time may indicate that a data center or a cloud resource is overloaded (Suakanto et al., 2012). Therefore, to ensure better performance of the cloud, tasks or jobs should be distributed to the most appropriate DC (service broker) and VM(s) (load balancing) to be executed with minimum response times (Iosup et al., 2011). Therefore, a minimum response time indicates an efficient execution time (i.e. maximum number of jobs to be performed per unit of time). Hence, the overall performance of the datacenter is also enhanced and not overloaded yet. Given that the network latency is proportional to the response time, it can be in some cases higher than the processing time itself. For instance, small and critical jobs may require very low processing time and the least response time, which can be achieved by selecting the network path with the least network latency. However, if the same path is selected repeatedly for all user requests within a specific location based on the proximity brokerage, the network path will soon become congested. Consequently, a congested link will have a direct effect on the available bandwidth, especially if jobs with big data need to be transmitted from the users to the DC and vice versa. It is also worth mentioning that the targeted DC will have many jobs waiting to be served which will significantly increase the overall response time as well. Therefore, the proposed service broker policy selects the data center based on the job size, the expected processing time, the network latency and the available bandwidth to minimize the overall response and processing time. The above factors are taken into consideration in order to come up with the finest routing policy through calculating the transmission time (i.e. network transfer delay) needed to transfer the user request based on the request size, available bandwidth and the network latency. Moreover, the proposed policy will have an estimate of the expected processing time, especially that most of the previous policies from the literature attempted to estimate the processing time based on the DC specifications, such as CPU, RAM, queuing time and VM configuration using complicated hypothetical equations to produce weighted values. These values usually are misleading because they are built based on the DC's resources and not the actual free resources which are changing constantly. However, this is out of the broker functionality scope because such functionality is the responsibility of the DC load balancer.

As a result, the proposed policy accommodates the current needs by taking real-time values to calculate the processing time to minimize the time needed to make the forwarding decision by the broker. Note that the job processing time can vary depending on the computational task to be performed. For instance, a smaller job requires less processing time if there was no I/O operation involved. However, since it is not the service broker functionality to analyze the jobs and determine their complexity, we considered the size of the job as an indication to the needed processing time with a positive relation between them.

In the proposed policy, all the DCs are initially assumed to have zero processing time, which is the least possible value. The DCs are then nominated and selected based on their proximity and the available bandwidth on the network path. After sending the very first job with a known size to a certain DC and the job results is received back from the DC, the time taken by the DC to process the job (processing time) is calculated based on Eq. (1)

$$P_T = R_T - (2 \times Nt_d) \tag{1}$$

where (P_T) is the processing time, (R_t) is the response time and the (Nt_d) is the network transfer delay. The network transfer delay (Nt_d) of a job with a size of (s) from the broker to a DC over a network path that has a delay value of (y) and an available bandwidth of (b) can be calculated using Eq. (2)

$$Nt_d = y + \frac{s}{b} \tag{2}$$

Having the processing time already known for the previous job, we can conclude the processing power available through Eq. (3) and the expected processing time for the next job as in Eq. (4)

$$P = \frac{s}{P_T} \tag{3}$$

where (P) is the expected processing time for the next job, (s) is the job size, and (P_T) is the processing time of the current job.

$$E(P) = \frac{s}{P} \tag{4}$$

The last assigned job processing time can also be a good indication of the DC availability. This is the way the performance optimized routing policy works, because a DC that can process a job in less time has higher availability chances as well as less jobs queued to be processed. However, this is not enough because the previous job could have a different complexity. Moreover, building the assumption on a previous job could lead to a misleading decision. Therefore, using the same logic to estimate the current job execution time will be more accurate, especially when applying the same scale to all jobs and DCs.

We implemented a simulated environment for the proposed broker policy called Variable Service Broker Routing Policy – VSBRP using the CloudSim environment. The proposed policy modifies the behavior of the original proximity routing policy through adding different parameters to enhance the response and processing time as portrayed in Fig. 2.

The following steps demonstrate how the proposed algorithm (Algorithm A) works:

1. Originally, the data centers are sorted in ascending order based on the delay matrix, thus selecting the least delay data centers. However, we sorted the data centers based on the delay and bandwidth matrices between UBs and DCs based on their availability; we came up with a composite value of delay and bandwidth and called it availability ratio (AV) as in Eq. (5).

$$\mathrm{AV}(\mathrm{DC}_i) = \frac{D[i,j]}{B[i,j]} \tag{5}$$

where i, is the ith DC, j is the user region, D is the delay matrix, and B is the bandwidth matrix. After determining each DC availability ratio, we added and sorted them in ascending order based on their availability ratio.

2. Initialize a list of DCs called "DC_Proccessing_Power", this list will map each DC to its processing power. The processing power is changing dynamically according to Eqs. (1), (3) and (4).
3. Finding the best DC with the least time needed to transfer the job over the underlying infrastructure and the least processing time as follows:
   a. Comparing the network transfer delay (**nwDelay**) for all data centers with the closest/selected DC (min(DC_{list})). The (**nwDelay**) is obtained from the following function call using the CloudSim *APIs inter netCharacteristics.getTotalDelay(src, DC, Req_Size)*
   b. The request data size (**Req_Size**) is variable and can be dynamically determined from the user request.
   c. If (**Req_Size**) is bigger than 10 kb.
      i. Add the expected (**nwDelay**) to the expected processing time for all DCs.
      ii. Select the DC with the lowest (**nwDelay**) and expected processing time.
   d. Else
      i. Select the closest DC if not the least availability.

Algorithm A:Variable Service Broker Routing Policy — **Complexity Analysis**

Input: R_n, UB_i, Req_size

Output: Destination DC name (DC_{name})

$\forall\ R_n,\ DC_{no} = \|DC_n\| > 1,\ n \in [0,1,...,6]$	..... t(n)
$DC_{ind} \leftarrow \text{index}\ (R_n, DC_n)$	
$DC_{list} = \text{prox.list} = \text{Ascend}\left(\frac{D_n[\ ]}{B_n[\ ]}\right)$	..... nlog(n)
if $DC_{list} \neq \text{null}$ then	..... t(1)
if $DC_{no} = 1$	
$DC_{Temp} \leftarrow DC_{list}[0]$	..... t(1)
else	
$DC_{Temp} \leftarrow \min\ (DC_{list}[\ ])$	..... t(n-1)
end if	
end if	
$\forall DC$	
$NT_{delay}[\ ] \leftarrow \text{GetTotalDelay}(UB_i, DC_m, Req\_size)$	..... t(1)
$DC_{nm} \leftarrow \text{Compare\_get\_min}(DC_{Temp}, NT_{delay}[\ ])$	..... t(n-1)
if $\|DC_{nm}\| = 1$	..... t(1)
$DC_{name} \leftarrow DC_{nm}[0]$	
else	
$DC_{name} \leftarrow \text{loadBalance}(DC_{nm})$	
end if	
return DC_{name}	

The running time of the proposed algorithm is $O(n(\log(n)))$. The algorithm is bounded by the sorting which can be achieved in $O(n(\log(n)))$ time using efficient sorting technique (i.e. quick sorting). However, the proposed method may achieve better running time through selecting the type of efficient sorting technique(s) that has a running time less than the quick sorting algorithm. The rest of the algorithm takes $O(n)$ time. So, the overall complexity is $O(n(\log(n)))$ as follows:

$$T(n) = t(n) + n(\log(n)) + t(1) + t(1) + t(n-1) + t(1) + t(n-1) + t(1) = n(\log(n))$$

4. Simulation and results

For the purpose of evaluating the proposed policy, we have used the Cloud-Analyst to implement the Variable Service Broker Routing Policy – VSBRP, and compare its performance with the existing routing policies, namely, the service proximity policy, performance optimized routing policy. However, before starting the simulation we have fixed the network delay and bandwidth matrices for all experiments as illustrated in Tables 2 and 3.

Four testing scenarios (see Section 4.1) were configured considering different situations such as varying loads, and users' needs as in Table 4. Each scenario has a set of user bases (UB) and data centers (DC) under the same Virtual Machine (VM) and network configurations. Each scenario was built with a number of DCs configurations as in Tables 5 and 7. Similarly, each scenario has a number of UB with the properties shown in Table 6. Moreover, the peak and off peak hours are considered to avoid any impact on the response time due to the changing loads on the network and the DCs.

All experiments were conducted over a 1-day period in order to get the most accurate results for off/in peak hours. Each experiment is carried out to evaluate the proposed policy response time, processing time and overall cost in comparison to the other policies within the cloud analyst environment. The obtained results for the response time is portrayed in Fig. 3, which shows that the proposed policy yields better response

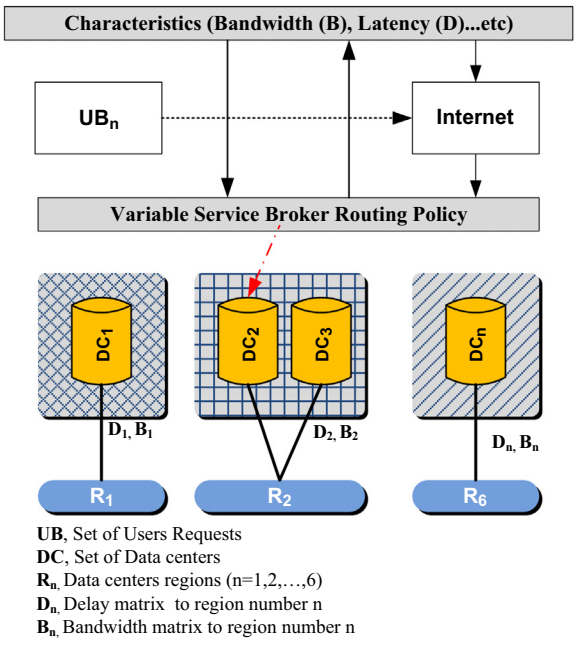

Figure 2 Variable Service Broker Routing Policy.

time compared to the proximity routing policy "Closest DC" and the performance optimized routing policy "Optimize Response Time". We were able to achieve significant improvements of response time in some cases like scenarios 1 and 2 because the proposed policy forwards the users' requests to the least congested network paths and servers with the minimal load. However, in other cases the response time was slightly reduced compared to the performance optimized routing policy because of the heavy load on the network and servers (i.e. big size requests). This is the worst case scenario the proposed policy may face, but still achieves better results compared to the existing policies.

As for the processing time, we measure the average processing time of the three routing policies. The proposed policy mostly provides better processing time compared to the other two routing policies as depicted in Fig. 4.

The proposed policy distributes the tasks in a balanced way based on the size of the request and the DC's availability with an average delay of (781.26) millisecond. Alternatively, we noticed that the overall cost of the proposed policy is almost the same as the three existing routing policies within the Cloud-Analyst simulator as depicted in Table 8 and Fig. 5.

Table 2 Delay matrix, transmission delay between regions in milliseconds.

Region	R0	R1	R2	R3	R4	R5
R0	25	100	150	250	250	100
R1	100	25	250	500	350	200
R2	150	250	25	150	150	200
R3	250	500	150	25	500	500
R4	250	350	150	500	25	500
R5	100	200	200	500	500	25

Table 3 Bandwidth matrix, available bandwidth between regions in Mbps.

Region	R0	R1	R2	R3	R4	R5
R0	2000	1000	1000	1000	1000	1000
R1	1000	800	1000	1000	1000	1000
R2	1000	1000	2500	1000	1000	1000
R3	1000	1000	1000	1500	1000	1000
R4	1000	1000	1000	1000	500	1000
R5	1000	1000	1000	1000	1000	2000

Table 4 Scenario description.

Scenario one (Section 4.1)	DCs are located at all regions. One UB is requesting services from one of these regions
Scenario two (Section 4.2)	DCs are located at all regions with three UBs requesting services from three different regions
Scenario three (Section 4.3)	Five data centers, two within the same region and three UB requesting services, each from its own region
Scenario four (Section 4.4)	DCs are located at all regions and UBs requesting services from all regions

The cost of the three routing policies in the four scenarios above are approximately equal in some cases such as scenarios 3 and 4 where for the proposed and the optimize response time-performance policies the cost is higher than the closest DC-proximity policy. The reason is that, the proposed and the optimize response time-performance policies invoke more VMs on different DCs while the Closest DC-proximity policy uses the same DC as long as possible, hence, invoking less VMs. The proposed policy selection is based on data center network availability for big data processing. The small increase in the overall cost can be traded by the improved results in processing and response time as illustrated in Table 8. The detailed results of the conducted experiments for the four scenarios are consolidated in Table 9.

Furthermore, we've evaluated the proposed policy under the same environment of the reconfigure dynamically peak time policy, proposed by Rekha and Dakshayini (2014). The environment details are illustrated in Tables 10 and 11. The obtained results are portrayed in Fig. 6.

From Fig. 6(a), we can notice that the response time of the proposed policy is better compared to the other policies. As for the cost, the proposed policy yields an overall cost of (3.21$) which is similar to the closest DC and the Optimize Response Time policies and significantly less than the Reconfigure Dynamically Peak Time policy, which yields an overall cost of (8.75$) because it routes user requests to data centers located at different geographical locations during off peak hours to improve the processing time. On the other hand, the proposed policy routes the user requests based on data center network availability as discussed earlier. Therefore, the processing time of the proposed policy is still better than the Reconfigure Dynamically Peak Time policy as shown in Fig. 6(b). The proposed policy yields a processing time of (0.77 ms) against (0.85 ms) of the Reconfigure Dynamically

Table 5 Data centers configurations and their physical hardware details.

Name	Region	Arch	OS	VMM	Cost per VM $/Hr	Memory cost $/s	Storage Cost $/s	Data transfer cost $/Gb	Physical HW unites
DC 1	0	x86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC 2	1	x86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC 3	2	x86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC 4	3	x86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC 5	4	x86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC 6	5	x86	Linux	Xen	0.1	0.05	0.1	0.1	1

Physical hardware details of DCs

ID	Memory (MB)	Storage (MB)	Available BW	Number of processors	Processor speed	VM policy
1	204,800	100,000,000	1,000,000	4	10,000	TIME_SHARED

Table 6 User base properties.

Scenario	Number of UB's	Regions	Requests user/hour	Request size (KB)	Peak hours start GMT	Peak hours end GMT	Avg peak users	Avg off peak users
Scenario 1	1	0	60	20	3	9	1000	100
Scenario 2	3	2,4,5	60	20,30,40	3	9	1000	100
Scenario 3	4	1,2,5	60	1,100,20,20	3	9	1000	100
Scenario 4	6	0–5	60	1,10,20,30,40	3	9	1000	100

Table 7 Load balancing and grouping factor configuration.

User grouping factors in user bases Equivalent to a number of simultaneous users from a single user base.	10
Request grouping factors in data centers Equivalent to number of simultaneous requests a single application server instance can support	10
Executable instruction length per request (bytes)	102,400
Load Balancing Policy across VMs in a single Data Center	Round Robin

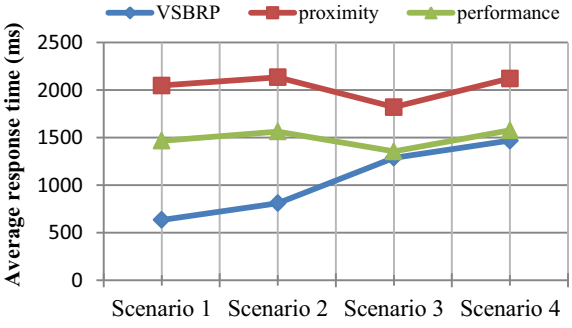

Figure 3 Comparison of different service broker policies response time.

Peak Time policy because, this policy attempts to share the load of a data center with other data centers when the original data center is busy.

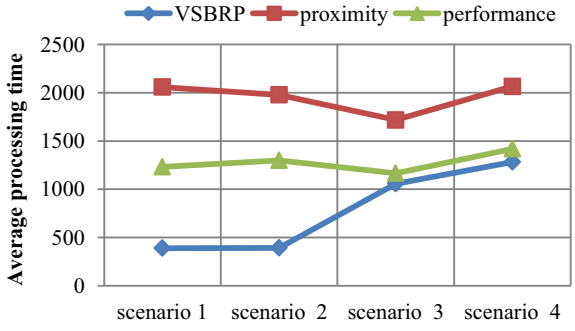

Figure 4 Comparison of different service broker policies processing time.

Table 8 Different broker policies cost, processing time, VM and data transfer averages.

Service broker policy	Avg response time	Avg processing time	Avg VM cost $	Avg data transfer cost $
VSBRP	1141.672	878.3331	69.0125	3084.305
Proximity	2032.898	1955.834	63.6125	3084.305
Performance	1487.82	1276.445	69.0125	3084.295

In order to reach an acceptable user satisfaction and resource utilization in cloud computing infrastructure, a competent and reasonable allocation of the computing resource has to be ensured. Therefore, selecting the proper service broker policy along with the proper VM allocation policy is vital to the overall cloud QoS (Wickremasinghe et al., 2010; Arora

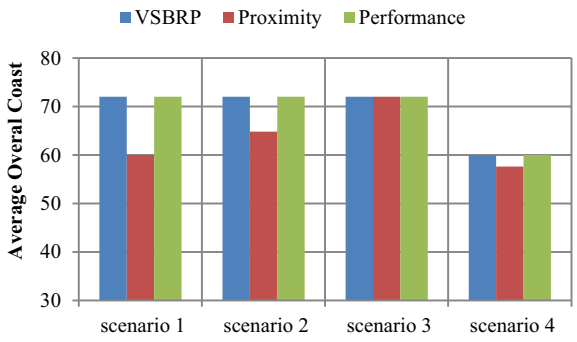

Figure 5 Comparison of different service broker policies overall cost.

Table 9 Experiments detailed results.

Service broker policy	Overall response time	Processing time	Total VM cost $	Total data transfer cost $
Scenario 1				
Proposed	635.37	389.85	72.02	973.53
Proximity	2049.1125	2058.18	60.02	973.53
Performance	1467.43	1233.27	72.02	973.53
Scenario 2				
Proposed	811.73	393.17	72.01	4388.28
Proximity	2135.04	1978.2175	64.81	4388.28
Performance	1564.03	1299.37	72.01	4388.25
Scenario 3				
Proposed	1288.25	1056.92	60.01	1998.86
Closest DC	1821.41	1718.13	57.61	1998.86
Performance	1355.76	1165.71	60.01	6900.35
Scenario 4				
Proposed	1469.31	1282.91	72.01	4976.55
Closest DC	2122.79	2065.55	72.01	4976.55
Performance	1577.00	1419.46	72.01	4976.54

Table 10 User base properties for the reconfigure dynamically peak time policy.

User Base	Region	Peak hour start	Peak hour end	Avg. peak users	Avg. off peak users
UB1	0	10	15	1000	100
UB2	1	10	15	1000	100
UB3	4	10	15	1000	100
UB4	3	10	15	1000	100
UB5	2	10	15	1000	100

and Tyagi, 2014; Zia and Khan, 2013). Therefore, we further evaluated the proposed VSBRP algorithm along with the three VM load balancing policies, namely, (1) Round Robin Policy in which the users' requests are handled in a circular manner on a FCF bases (Shah et al., 2013; Goyal, 2014; Limbani and Oza, 2012). (2) Throttled Policy, in which each VM is assigned only one job at a time and another job can be assigned only when the current job has completed successfully

Table 11 Data Centers Configurations for the reconfigure dynamically peak time policy.

DC	Number of VM's	Region	Cost per VM/hour	Data transfer cost
DC1	5	0	0.1	1
DC2	50	1	0.1	1
DC3	25	2	0.1	1
DC4	100	3	0.1	1

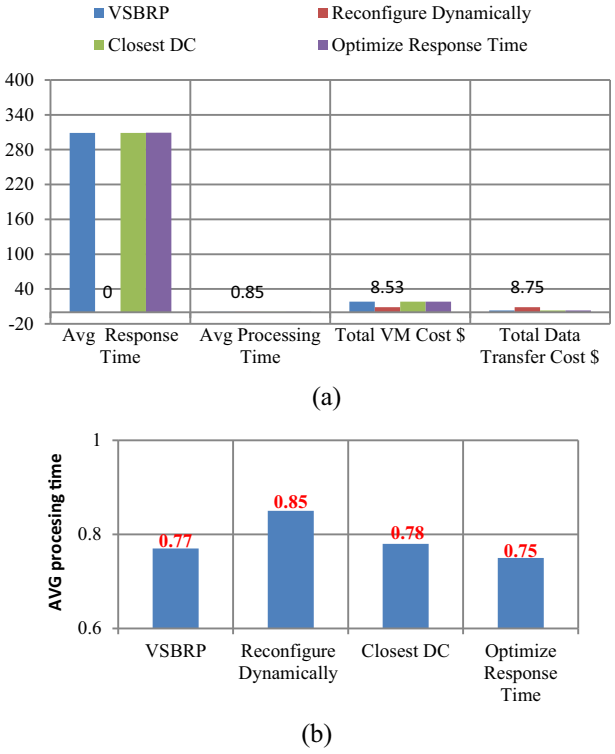

Figure 6 Comparison of different service broker policies on overall cost, response time (a) and processing time (b).

or the request will be queued until any VM became available (Goyal, 2014; Domanal and Reddy, 2013). (3) Active Monitoring Policy (i.e. Equally Spread Execution Load) distributed the load equally among all the VMs by actively monitoring their load (Goyal, 2014; Domanal and Reddy, 2014). The purpose of such evaluation is to identify the best VM load balancing policy that can further enhance the response and processing time through distributing the task to the most suitable VM for execution (Limbani and Oza, 2012).

For the above purpose, we repeated the above experiment under the same environment with different VM allocation policies, the obtained results are demonstrated in Table 12.

The result analysis demonstrated in Table 12 reveals that out of the three considered VM Load Balancing Policies, the overall response and processing time of the Datacenter is better in the case of Throttled Load balancing Policy. In fact, that is because the proposed VSBRP policy guarantees an efficient distribution of the users' requests to all data centers that are available and ready for the job. While the Throttled load balancing policy ensures that each VM has a suitable one job to

Table 12 Evaluation results of applying VSBRP along with various LB Policy at VM level.

Load balancing policy	Average response time	DC processing time	Total VM cost $	Total data transfer cost $
Scenario 1				
Round Robin	635.37	389.85	72.02	973.53
Active Monitoring	635.37	389.85	72.02	973.53
Throttled	498.93	252.95	72.02	973.53
Scenario 2				
Round Robin	811.73	393.17	72.01	4388.28
Active Monitoring	811.73	393.17	72.01	4388.28
Throttled	673.71	251.95	72.01	4388.28
Scenario 3				
Round Robin	1288.25	1056.92	72.01	4976.55
Active Monitoring	1288.25	1056.92	72.01	4976.55
Throttled	806.85	577.24	72.01	4976.55
Scenario 4				
Round Robin	1469.31	1282.91	60.01	1998.86
Active Monitoring	1469.31	1282.91	60.01	1998.86
Throttled	860.36	675.83	60.01	1998.86

carry out. This fair distribution significantly improves the response and processing time using the proposed policy.

4.1. Testing scenarios

In this section we will discuss and illustrate the conducted experiments and to show and rationalize the obtained results in Section 4.

4.1.1. Experiment one, testing scenario 1

In this experiment, the logically expected best response time is the proximity routing policy since there is one UB and the closest DC would be the most efficient one as in Fig. 7. The line between DC1 and UB1 represents the DC selection. However, this is not the case, we get the best response time using the proposed service broker policy because it depends on the DC availability ratio (Eqs. (1) and (3)), which can be justified simply by considering the fact that one DC will be overloaded by user requests and significantly increase the time needed to finish each job causing more delays.

Moreover, the available bandwidth rapidly decreases especially during peak hours, which requires more time to transfer the job to the selected DC and back to the user. While in the proposed policy, the workload distribution considers the load on the communication path and the DC to be selected, which results in forwarding user's requests to multiple DC as in Fig. 8, which achieves better load balancing and great reduction in the overall response time. On the other hand, the performance optimized routing policy considers only the server load, neglecting the network availability and the impact of the job size, which resulted in higher response time.

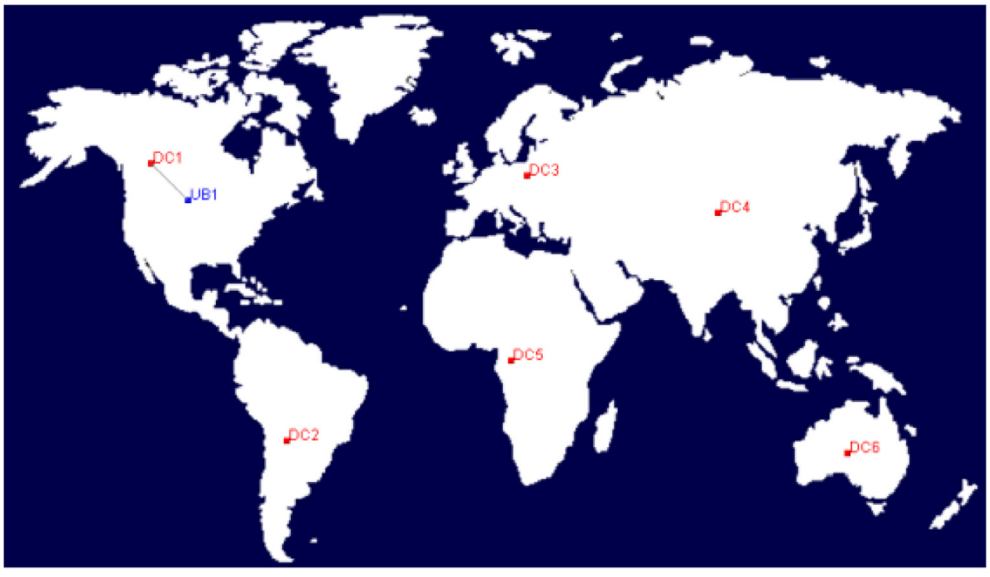

Figure 7 Experiment 1 (proximity routing policy DC selection).

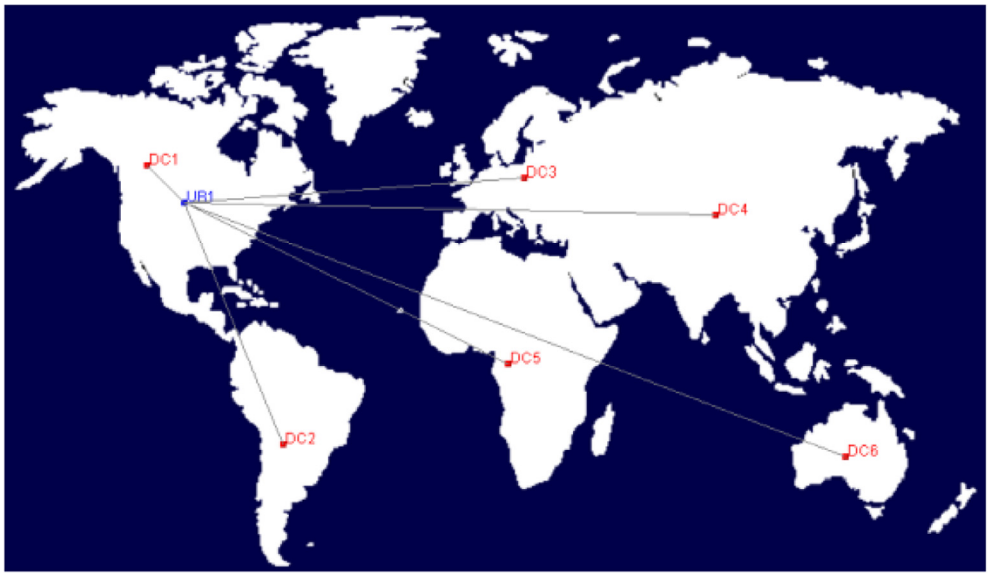

Figure 8 Experiment 1 (proposed policy DC selection).

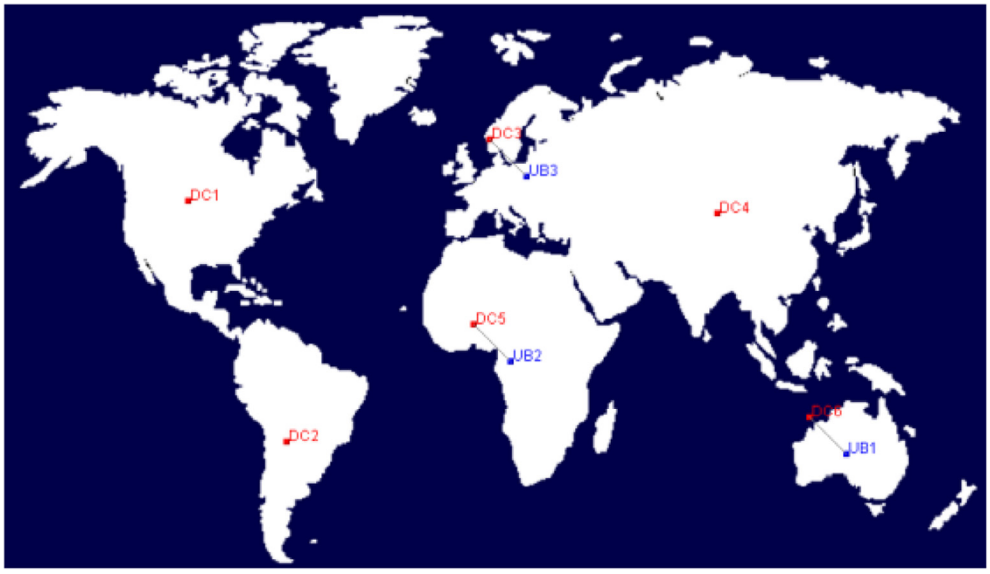

Figure 9 Experiment 2 (proximity routing policy DC selection).

4.2. Experiment two, testing scenario 2

In this experiment the proposed policy achieves better results than the other two policies because of the load balancing. Fig. 9 shows that the proximity routing policy always selects the closest DC to the UB, which will also cause higher response time. While the performance optimized routing policy balances the loads between the available DCs but it doesn't consider the state of the network and the impact of the request size, it only considers the previous DC load.

4.3. Experiment three, testing scenario 3

This scenario was conducted to show the other routing polices' drawbacks, which is leaving DCs idle and unselected as in Fig. 10. If the DCs have higher network delay even though they reduce the response time and the job distribution over all available DCs, they will not always yield a better response time. However, the reduction in response time in the proposed policy was relatively less than the previous scenarios but still better than the other routing policies because of the same reasons stated in scenarios 1 and 2.

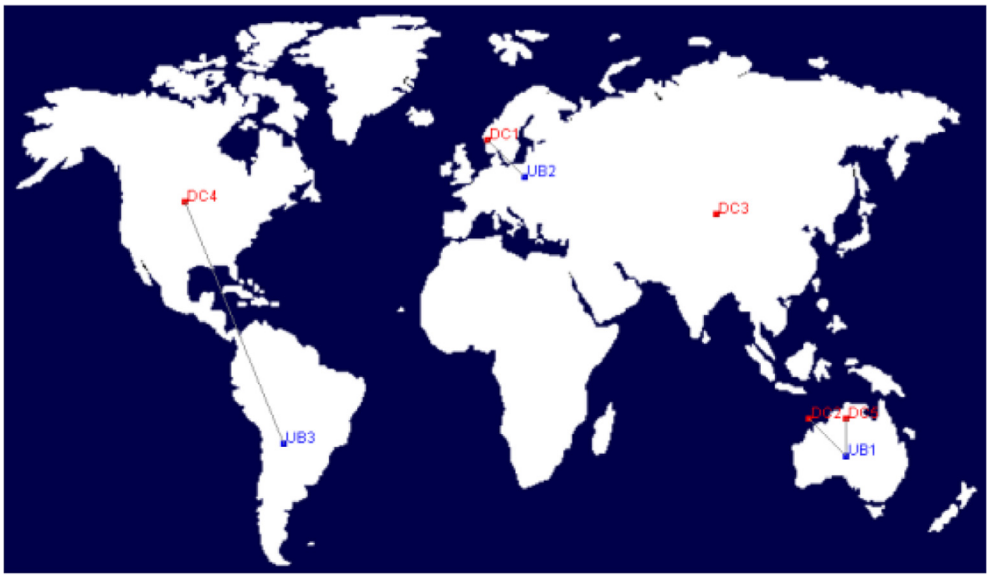

Figure 10 Experiment 3 (proximity routing policy DC selection).

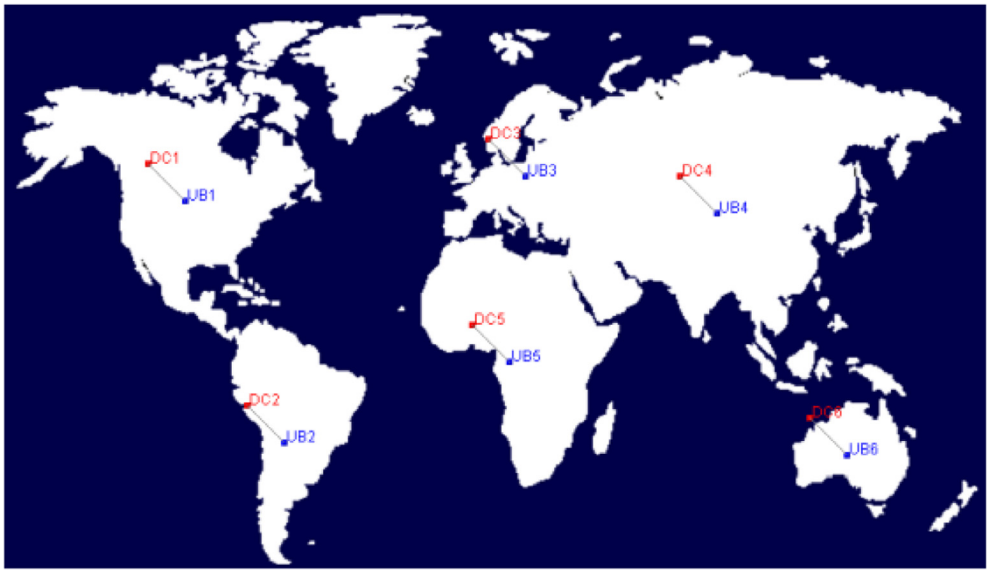

Figure 11 Experiment 4 (proximity routing policy DC selection).

4.4. Experiment four, testing scenario 4

This scenario shows and proves the need for an optimal and variable routing policy that can accommodate different situations by distributing jobs on different DCs only as needed. Selecting the closest DC isn't enough as in Fig. 11. On the other hand, too much distribution will not always be the best solution to the problem as illustrated in Fig. 8.

Fig. 12 shows that the optimized routing policy balances the loads between all available DCs by giving higher priority to selecting the least loaded DC and not considering the effect of choosing a DC with high latency. This is a clear draw back on the performance that may increase the response time.

In Fig. 13, the proposed policy is obviously choosing less data centers to forward the jobs and will only choose a new DC if necessary. This will reduce the response time.

5. Conclusion

In this paper, we presented a Variable Service Broker Routing Policy to minimize the processing and response time of user's requests within an acceptable range of cost. The proposed policy modifies two behavior of the proximity routing policy to select the data centers in an efficient way. The proposed policy modifies the sorting and selection equations of the old policy. The first modification is to incorporate the bandwidth into the

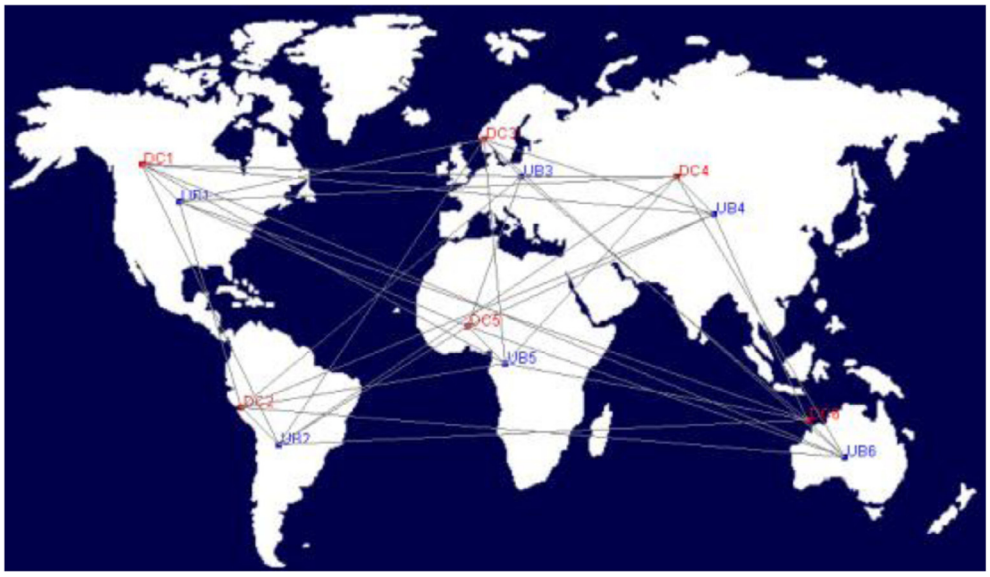

Figure 12 Experiment 4 (optimized routing policy).

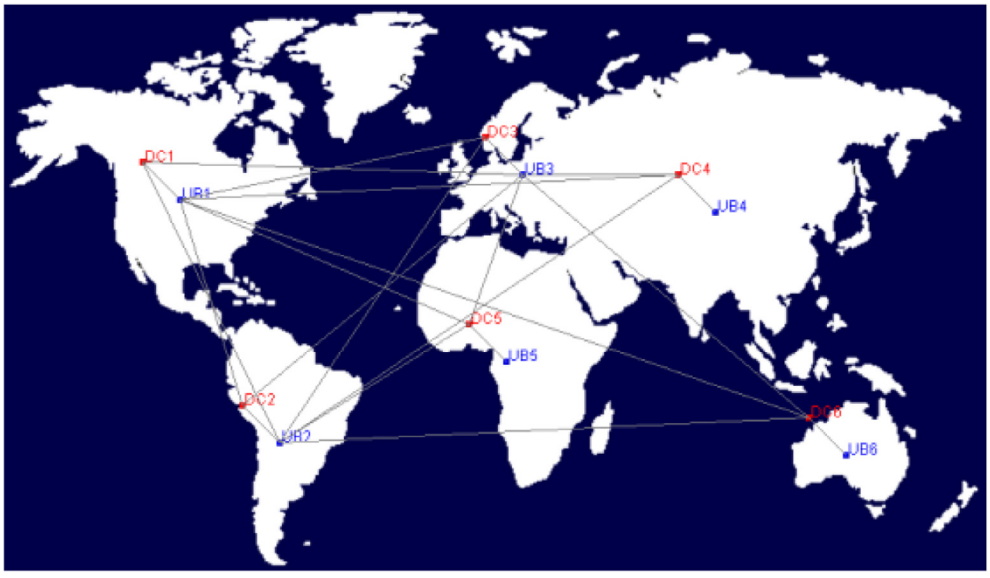

Figure 13 Experiment 4 (proposed policy DC selection).

proximity routing policy selection rather than relaying only on the delay factor. This modification enhances the response time with regards to the request size. On the other hand, adding the request size into the sorting equation enables the proposed policy to work with variable requests size to calculate network delay. Finally, the proposed policy selects the DC with the least delay considering the request size, real time available bandwidth, network delay and the expected processing time of the current job. This makes it different from the performance optimized routing policy which relies on the least and last processing time of DC with no consideration for the job size. The proposed policy response time and processing time is improved compared to other known policies within the Cloud-Analyst simulator. The proposed policy is evaluated and compared with existing policies using the CloudAnalyst simulator. The simulation experiment results show a noticeable improvement in the average overall response and processing time. Furthermore, the simulation experiment results show that the proposed policy can perform better if it adopts the Throttled load balancing policy.

6. Future work

Improving the financial cost and power consumption is still to be researched and improved if possible. The proposed policy requires further improvements especially in case of variable DC's cost or power consumptions, which could be an important factor in some cases and might be preferred over efficiency and speed. So we are looking for improving the proposed algorithm by counting for cost and power. On the other hand,

intelligent based approaches can be adopted to optimize the route selection to minimize the overall coast as an objective. As well as incorporating the DC efficiency in the brokerage policy to select an optimized route based on the above parameters.

References

Ahmed, S., 2012. Enhanced Proximity-based Routing Policy for Service Brokering in Cloud Computing. Int. J. Eng. Res. Appl. (India) 2 (2), 1453–1455.

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., et al, 2010. A view of cloud computing. Commun. ACM 53 (4), 50–58.

Arora, V., Tyagi, S.S., 2014. Performance evaluation of load balancing policies across virtual machines in a data center. In: Paper Presented at the Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on 6–8 Feb. 2014.

Beloglazov, A., Buyya, R., 2013. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. IEEE Trans. Parallel Distrib. Syst. 24 (7), 1366–1379.

Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R., 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Pract. Exp. 41 (1), 23–50. http://dx.doi.org/10.1002/spe.995.

Dillon, T., Wu, C., Chang, E., 2010. Cloud computing: issues and challenges. Paper presented at the Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on.

Dinh, H.T., Lee, C., Niyato, D., Wang, P., 2013. A survey of mobile cloud computing: architecture, applications, and approaches. Wireless Commun. Mobile Comput. 13 (18), 1587–1611. http://dx.doi.org/10.1002/wcm.1203.

Domanal, S.G., Reddy, G.R.M., 2013. Load balancing in cloud computing using modified throttled algorithm. Paper presented at the Cloud Computing in Emerging Markets (CCEM), 2013 IEEE International Conference on.

Domanal, S.G., Reddy, G.R.M., 2014. Optimal load balancing in cloud computing by efficient utilization of virtual machines. Paper presented at the Communication Systems and Networks (COMSNETS), 2014 Sixth International Conference on.

Goyal, A., 2014. A study of load balancing in cloud computing using soft computing techniques. Int. J. Comput. Appl. 92 (9).

Houidi, I., Mechtri, M., Louati, W., Zeghlache, D., 2011. Cloud service delivery across multiple cloud platforms. Paper presented at the Services Computing (SCC), 2011 IEEE International Conference on.

Iosup, A., Ostermann, S., Yigitbasi, M.N., Prodan, R., Fahringer, T., Epema, D.H., 2011. Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Trans. Parallel Distrib. Syst. 22 (6), 931–945.

Kapgate, D., 2014. Efficient service broker algorithm for data center selection in cloud computing. Int. J. Comput. Sci. Mobile Comput. 3 (1).

Kremer, J., 2013. Virtualization and Cloud Computing, Steps in the Evolution from Virtualization to Private Cloud Infrastructure as a Service White paper on virtualization, USA.

Limbani, D., Oza, B., 2012. A proposed service broker strategy in cloudanalyst for cost-effective data center selection. Int. J. Eng. Res. Appl. (India) 2 (1), 793–797.

Limbani, D., Oza, B., 2012. A proposed service broker policy for data center selection in cloud environment with implementation. Int. J. Comput. Technol. Appl. 3 (3).

Mell, P., Grance, T., 2009. The NIST Definition of Cloud Computing. National Institute of Standards and Technology 53 (6), 50.

Mishra, R.K., Bhukya, S.N., 2014. Service broker algorithm for cloud-analyst. Int. J. Comput. Sci. Inf. Technol. 5 (3), 3957–3962.

Mishra, R.K., Kumar, S., Sreenu Naik, B., 2014. Priority based round-robin service broker algorithm for cloud-analyst. In: Paper Presented at the Advance Computing Conference (IACC), 2014 IEEE International 21–22 Feb. 2014.

Rajesh, G., Sreenivasulu, G., 2014. The issues of cloud service delivery through virtualization of Dynamically Generated multiple virtual machine Services without missing deadline on the World Wide Web. Int. J. Curr. Eng. Tech. 4 (4), 2758–2762.

Rani, P., Chauhan, R., Chauhan, R., 2015. An enhancement in service broker policy for cloud-analyst. Int. J. Comput. Appl. 115 (12).

Rekha, P.M., Dakshayini, M., 2014. Cost based data center selection policy for large scale networks. In: Paper presented at the Computation of Power, Energy, Information and Communication (ICCPEIC), 2014 International Conference on 16–17 April 2014.

Rimal, B.P., Choi, E., Lumb, I., 2009. A taxonomy and survey of cloud computing systems. Paper presented at the INC, IMS and IDC, 2009. In: NCM'09. Fifth International Joint Conference on.

Semwal, A., Rawat, P., 2014. Performance evaluation of cloud application with constant data center configuration and variable service broker policy using CloudSim. IJERSTE.

Shah, M.M.D., Kariyani, M.A.A., Agrawal, M.D.L., 2013. Allocation of virtual machines in cloud computing using load balancing algorithm. Int. J. Comput. Sci. Inf. Technol. Secur. (IJCSITS), ISSN: 2249–9555.

Sharma, V., 2014. Efficient data center selection policy for service proximity service broker in CloudAnalyst. Int. J. Innovative Comp. Sci. Eng. (IJICSE) 1 (1), 21–28.

Sharma, V., Rathi, R., Bola, S.K., 2013. Round-robin data center selection in single region for service proximity service broker in CloudAnalyst. Int. J. Comput. Technol. 4 (2a1), 254–260.

Suakanto, S., Supangkat, S.H., Saragih, R., 2012. Performance Measurement of Cloud Computing Services. arXiv preprint arXiv:1205.1622.

Wickremasinghe, B., Calheiros, R.N., Buyya, R., 2010. CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications. In: Paper Presented at the Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on 20–23 April 2010.

Zhang, Q., Cheng, L., Boutaba, R., 2010. Cloud computing: state-of-the-art and research challenges. J. Internet Serv. Appl. 1 (1), 7–18.

Zia, A., Khan, M., 2013. A scheme to reduce response time in cloud computing environment. Int. J. Modern Educ. Comput. Sci. (IJMECS) 5 (6), 56.