Journal of Network and Computer Applications 75 (2016) 47–57

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Cost-aware service brokering and performance sentient load balancing algorithms in the cloud

Ranesh Kumar Naha*, Mohamed Othman

Department of Communication Technology and Network, Universiti Putra Malaysia, 43300 UPM Serdang, Selangor D.E., Malaysia

ARTICLE INFO

Article history:
Received 12 May 2015
Received in revised form
3 July 2016
Accepted 22 August 2016
Available online 24 August 2016

Keywords:
Cloud brokering
Load balancing
Cloud computing
Resource management

ABSTRACT

On-demand resource provisioning makes cloud computing a cutting edge technology. All cloud service providers offer computing resources with their own interface type, instance type, and pricing policy, among other service features. A cloud-based service broker provides intermediation to seek appropriate service providers in terms a suitable trade-off between price and performance. On the other hand, load balancing among cloud resources ensures efficient use of a physical infrastructure, and at the same time, minimizes execution time. This makes service brokers and load balancing among the most important issues in cloud computing systems. This paper aims to propose three different cloud brokering algorithms, and a load balancing algorithm. A simulation-based deployment confirms that our proposed algorithms minimized the cost, and at the same time, witnessed gains in service performance.

1. Introduction

The cloud consists of large data centers, or groups of large data centers, which may be located in one or multiple geographical regions where the cloud provides unlimited computing resources that are available to satisfy user demand. The cloud can be hosted by enterprises, governments and service providers (Bernstein et al., 2011, 2009). Content, storage and computing capable of providing services anywhere throughout the network are referred to as "Intercloud" (Bernstein and Vij, 2010). In an interoperability scenario, clouds must be able to detect each other to exchange information (Vecchiola et al., 2011). Cloud infrastructures are also represented by services that are not only used, but also installed, deployed or replicated, with the help of virtualization. These services are applied in complex business processes that further complicate the fulfillment of Service Level Agreements (SLAs). For example, due to the changing components, workloads, external conditions, hardware, and software failures, established SLAs may be violated. Frequent user interaction with the system during SLA negotiations and service executions (which are usually necessary in the case of failures) may pose a challenge to successful cloud Computing. Garg et al. (2014) propose a scheduling technique which considers SLA-based VM management with mix workload.

A promising use case of the 'Intercloud vision' was defined by Buyya et al. (2009), which involves market transactions via brokers. In such a use case, a broker entity is a mediator between the cloud consumer and multiple interoperable cloud providers, in order to support the former in selecting the provider, which better meets user requirements. Another value-added broker service is the easy deployment and management of a user's service, regardless of the selected provider, through a uniform interface. The evaluation of the broker on a real-world test bed is typically cost- and time-consuming, since a great volume of cloud resources is required to achieve realistic, reliable results. A more promising and cost-saving approach for the process of broker evaluation is the application of a simulation environment.

In this study, the researchers focused on load balancing among data centers and virtual machines; the limitations of efficient cloud load balancing motivated us to develop a novel load balancing algorithm. The proposed algorithm reduces overall processing and response times, since tasks are allocated to the available physical resources in an efficient manner. The proposed algorithm was found to outperform previously proposed algorithms. In this study, cloud brokering and load balancing algorithms for a cloud computing environment were proposed. A discussion of the simulation results of the three cloud brokering algorithms and the load balancing algorithm is also presented.

2. Related work

Many standard bodies are currently working to define common standards for cloud computing. These standards have been proposed by numerous cloud standard organizations. Cloud users

* Corresponding author.

http://dx.doi.org/10.1016/j.jnca.2016.08.018
1084-8045/© 2016 Elsevier Ltd. All rights reserved.

currently face the challenge to select the appropriate cloud in order to satisfy their specific requirements. Using an intermediate cloud brokering service to find a specific provider that satisfies their requirements is a promising research direction (Jrad et al., 2012). The two primary underlying components of cloud brokering are resource provisioning and scheduling. Van den Bossche et al. (2010) propose a scheduling approach for a hybrid cloud that performs operations in terms of scalability, cost minimization and feasibility. Through binary integer programming, their approach supports users in their decision making process with partial automatization. The researchers also claim that they addressed the resource management problem first in hybrid clouds using this technique.

Wickremasinghe et al. (2010) propose the CloudAnalyst tool in order to model and evaluate real-life problems deployed in the cloud, exploiting a case study based on a social networking application. In their work, they reveal the manner in which a cloud service broker optimizes system resources based on applications in various geographic locations. Moreover, they proposed three Virtual Machine (VM) load balancing algorithms and two cloud brokering algorithms for their experiments on the CloudAnalyst tool.

Rochwerger et al. (2011) show that, although there is a lack of interoperability, among other limitations, in present technologies, federated clouds have a vast future prospective. It has also been demonstrated that efficient resource scheduling and provisioning definitely increase cloud brokering performance.

Ferreto et al. (2011) worked on resource management, which extends existing solutions of server consolidation. Their proposed solution defines a constraint that defines VM with variable capacity to be migrated and do not migrate VM with steady usage in order to minimize the usage of the physical server. Ferreto et al. (2011) called this approach dynamic consolidation with migration control. Their results show that, with this migration control system, data centers would be greatly benefited.

Tordsson et al. (2012) propose a cloud brokering architecture for VM management in a hybrid cloud computing environment. Tordsson et al. (2012) also propose an application placement optimization algorithm with the application of integer programming formulations that facilitate a price performance trade-off. The Aneka (Vecchiola et al., 2011) software platform involves the management of provisioned resources from clusters, grids and clouds. The researchers prove that Aneka supports Quality of Service (QoS) aware execution of controlled applications in crossbreed clouds. Experimental results of their work suggest that Aneka has the ability to allocate resources proficiently with the intention of a reduction in application execution time.

Calheiros et al. (2012) propose a cloud coordinator architecture that works in private and public cloud environments. This cloud coordinator represents brokers and data centers in the InterCloud marketplace, and is liable for resource negotiation management and publishing offers. The researchers discuss the impact on elastic applications and the effectiveness of their proposed architecture in a modest scenario.

Quarati et al. (2013) present a cloud brokering algorithm based on various types of scheduling conditions for a hybrid cloud. Simulation results of their work manifest that this brokering algorithm maximizes broker revenues and user satisfaction. The researchers apply energy saving mechanisms in order to increase the broker's revenues. Akhter and Othman (2014) proposed VM provisioning in an energy efficient manner for a cloud data center, and incorporate various strategies (Akhter and Othman, 2016).

Kessaci et al. (2013) propose the Pareto Multi-Objective Genetic Algorithm for Cloud Brokering (MOGACB), which aims for both client satisfaction and the broker's profit. They mention that client satisfaction is mostly related to response time rather than on the cost of instances.

The objective of cloud load balancing is to speed up the execution time of applications. Numerous studies were performed on cloud load balancing during the past few years. Fang et al. (2010) discuss two level tasks scheduling for high resource utilization. Their proposed application and VM level task scheduling was proven by the CloudSim (Calheiros et al., 2011) toolkit simulation framework. However, factors such as bandwidth and cost were not considered in their work at the period of load balancing.

Randles et al. (2010) propose Biased Random Sampling, Foraging Behavior and Active Clustering load balancing, inspired by the honeybee load balancing technique. Their proposed algorithm was developed for a cloud environment with heterogeneous nodes, but was simulated on a small scale system. Round robin and throttling-based load balancing was proposed by Wickremasinghe et al. (2010). They studied application behavior in a large-scale distributed cloud computing environment by the use of simulations. However, load balancing performance could be further enhanced, which is shown in our proposed algorithm.

VM resource load balancing uses a genetic algorithm proposed by Hu et al. (2010). This algorithm addresses the load imbalance and over migration cost problems in traditional algorithms proposed in prior work. However, a management and monitoring mechanism is still required for the dynamic changes of VMs.

Wang et al. (2010) propose Opportunistic Load Balancing and Load Balance Min–Min, which are static scheduling algorithms for system load balancing. These algorithms were developed for three-level cloud computing networks. Request Manager, Service Manager and Service Node are the three hierarchical components in their framework. Opportunistic Load Balancing attempts to make all systems busy in operation time of load balancing, while the Load Balance Min–Min algorithm minimizes execution time.

Maguluri et al. (2012) propose a stochastic model for load balancing in cloud computing clusters, focusing on load balancing among servers. They demonstrate that widely used BestFit scheduling is not throughput-optimal. Based on various optimization criteria and user constraints, Lucas-Simarro et al. (2013) propose a load balancing and scheduling algorithm for multiple clouds. Their work demonstrated cost effectiveness and performance improvement, simultaneously, in their proposed algorithms.

Another honey bee inspired load balancing algorithm was proposed by Krishna (2013). This algorithm provides load balance across the VMs and maximizes throughput, and the amount of time of queued tasks was minimal, which leads to a significant improvement of average execution time. This work focuses on task priority rather than QoS factors.

Our initial work on load balancing was presented in Naha and Othman (2014). Round robin with server affinity was proposed by Mahajan et al. (2013). The authors improve the algorithms functionality by an efficient load balancing algorithm. However, application load for an hour may not lead to the actual results for these kinds of simulations, since it is not possible to realize the effect of peak and off peak hours load balancing without at least a one day workload. Load balancing using the firefly model was proposed by Florence and Shanthi (2014). The simulation of this work was conducted in a very small-scale cloud environment. VM assignment algorithms depending on load were proposed by Domanal and Reddy (2014). They determine algorithm response in the case of dynamic and mix workload. Sun et al. (2014) conducted a comprehensive analysis on the selection approach of cloud services, and identified potential research gaps on cloud service selection.

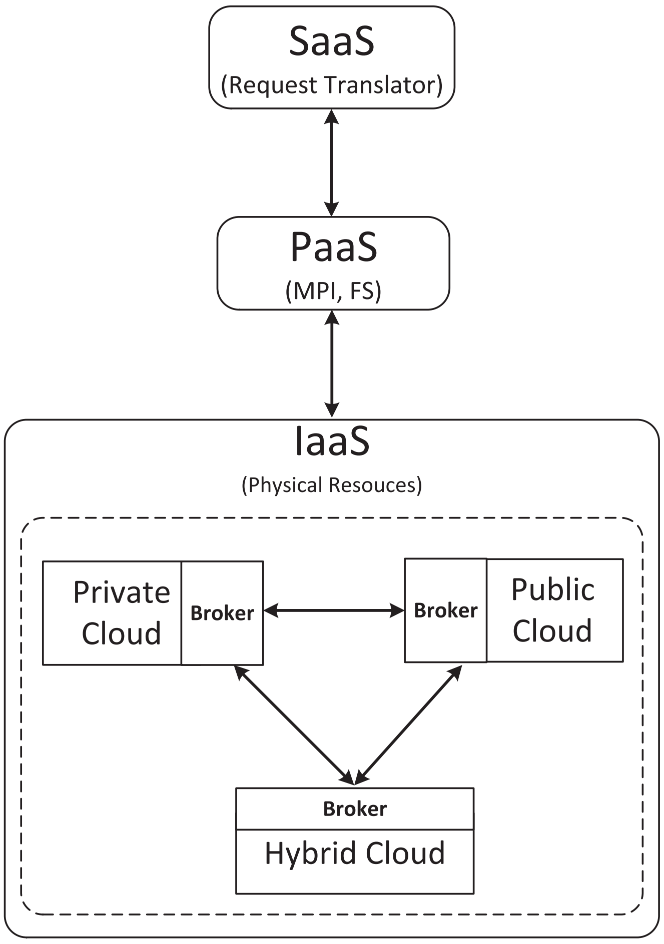

Fig. 1. Cloud service layered model.

3. Cloud service models

3.1. Cloud services layered model

IaaS, PaaS and SaaS are the three basic cloud service layers. Fig. 1 represents a cloud service layered model. The top layer is the SaaS layer, which deals with the requirements of application execution. The SaaS layer translates platform requirements for the application, and passes them to the PaaS layer.

The PaaS layer then provides a development library, runtime environment, scaling and layer decoupling. This layer acts through traditional middleware, and establishes a bridge between the infrastructure and the application requirements. It does not determine the actual infrastructure such as the number of VMs needed. However, it determines the higher representation of execution units, e.g., threads, process and tasks. Microsoft Azure and Google App Engine are some real world examples of PaaS cloud services.

The IaaS layer deals with number of processors, disk size, virtual machine and network connectivity. This layer determines the provisioning of virtual resources. Brokers of every cloud are concerned with the availability of system resources, and notify of resource costs and runtime performance to other brokers. Our proposed algorithms were designed for the brokers in the IaaS layer. Brokering in the SaaS and PaaS layers differ by fulfilling distinguished criteria. SaaS layer brokering is based on SLA and user requirements, while PaaS layer brokering is based on runtime and deployment support.

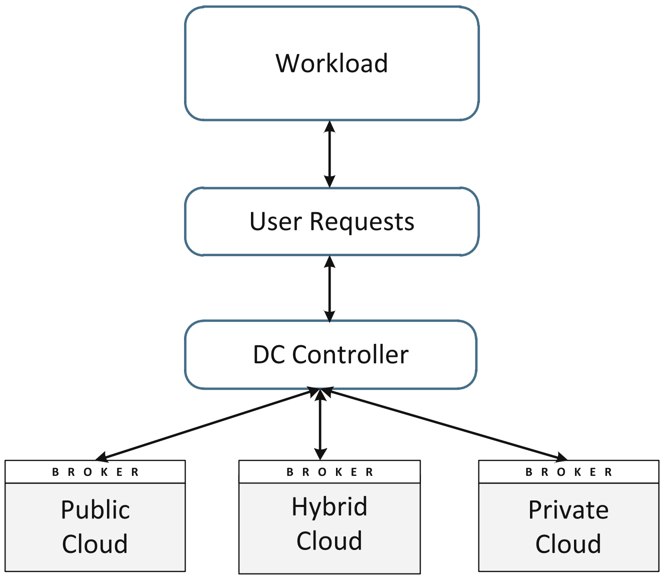

Fig. 2. Workload and resource request flow.

3.2. Workload and resource model

It is difficult to outline a realistic workload for an IaaS cloud. In real-world scenarios, an IaaS workload is dynamically changeable and does not have any fixed patterns. However, for simulation purposes, we can use workload traces or generated workloads. Resources are provisioned as the demand of the present workload. Fig. 2 depicts the workload and resources model. The workload is passed to the DC controller as a user request. Based on the request, the broker communicates with other resource brokers for resource availability. Among the available resources, the broker finds the best resources based on the users constraints. Once the broker finds the most suitable resources, it proceeds through allocation and provisioning policies.

4. Data transmission and user request processing

Data transmission delay (D_t) depends on network latency (N_l) and data transfer time (D_{tr}). Data transmission delay is measured in millisecond (ms). Network latency between regions is taken as a latency matrix, which is hypothetical data that follows a Poisson distribution. Data transfer time depends on the size of the data of a single request (R_S) and bandwidth per user (B_p). Therefore, data transfer time could be denoted as ($D_{tr} = \frac{R_S}{B_p}$); consequently, we can formulate the data transmission delay as the following equation:

$$D_t = N_l + \frac{R_s}{B_p} \tag{1}$$

Bandwidth per user is calculated from total available bandwidth (T_b) and concurrent user requests between two regions (R_n). Bandwidth per user is computed by the following equation:

$$B_p = \frac{T_b}{R_n} \tag{2}$$

Data center controller received requests from the user base and the data center controller divides this request into sub-cloudlets by Internet cloudlets. Next, the sub-cloudlet request is assigned to the VMs for further processing through the load balancer. Data center processing time is calculated by the receipt on the first response of the sub-cloudlet. So the data center processing time is the total time needed to process the user request until it sends the request to the VM. The time necessary to complete a single user request is considered the data center response time. We measured data center processing time and data center response time.

5. Proposed algorithms

5.1. Description of proposed cloud brokering algorithms

The cloud broker is responsible for handling provider resources and user requests. Primary work for cloud brokering was presented by Naha et al. (2015). To allocate resources according to user requirements, the broker must discover all available resources. Different users must have different constraints in order to allocate resources. On the other hand, each provider maintains their own SLA. A cloud broker always considers user satisfaction, while avoiding SLA violations. After discovering available resources, a cloud broker allocates resources based the user's specific task. The brokering performance depends on efficient resource discovery and allocation. Fig. 3 shows the queueing system of the proposed broker.

5.1.1. Cost aware brokering

We use two different policies to discover the available resources for users, along with a unique algorithm for each. The algorithm for resource selection over cost and load is shown in Algorithm 1. In this algorithm, regionalList is a list of regions in which data centers are located. The dcCostLoadList holds cost and load wise list for future allocation. The sm holds the dcName of the lowest cost. Cost-aware brokering algorithms always select the VM with the lowest cost. They also always update the list if an available lowest cost VM is found.

Algorithm 1. Resource discovery (cost and load) algorithm.

```
Input: userBase, regionalList
Output: dcName
1  Fetch region of the requested sender;
2  if $regionalList$ is not $NULL$ then
3      $listSize \leftarrow Size(regionalList)$;
4      if $listSize$ is 1 then
5          $dcName \leftarrow regionalList.get(0)$;
6      else
7          $dcCostLoadList \leftarrow regionalist$ sort by cost then by load;
8          for all $dcp$ in $dcCostLoadList$ do
9              if $dcCostLoadList.get(sm) > dcCostLoadList.get(dcp)$
                then
10                 $sm \leftarrow dcp$;
11             end
12         end
13         $dcName \leftarrow dcCostLoadList.get(sm)$;
14     end
15 end
16 Return $dcName$
```

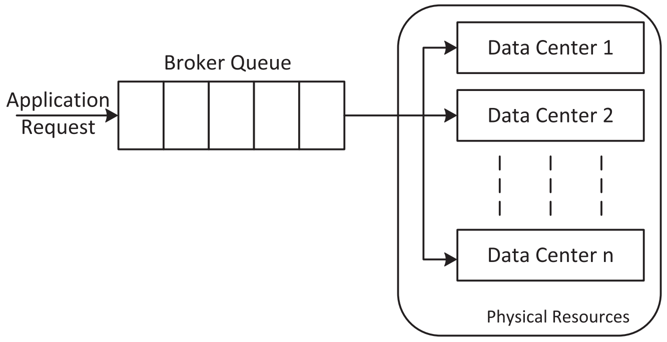

Fig. 3. Broker queueing process.

This algorithm maintains a list of data centers that is indexed by data center location. In the case a request is received from user bases by the broker, the broker retrieves the region of the sender. It then sends queries to determine which data centers are located in the same region. These data centers are listed in an array in terms of network latency.

Latency between the client and the data center is considered network latency. Later on, other data centers located in different regions are added to the same array in terms of network latency. If only one data center is found in the same region, then it selects the only data center in that particular region. In the final step, if more than one data center is found by the broker, it will select the data center having the minimum cost and minimum workload.

5.1.2. Load aware brokering algorithm

The load-aware algorithm distributes all requests among all available data centers. The resource selection uses a load balancing technique, as shown in Algorithm 2. The processing time is occasionally reduced if the jobs are distributed among various data centers. But this could increase operational costs.

Algorithm 2. Resource discovery (load aware) algorithm.

```
Input: userBase, regionalList
Output: dcName
Fetch region of the requested sender;
if $regionalList$ is not $NULL$ then
    $listSize \leftarrow Size(regionalList)$;
    if $listSize$ is 1 then
        $dcName \leftarrow regionalList.get(0)$;
    else
        getRandomNumber($listSize$) ;
    end
    $dcName \leftarrow dcCostLoadList.get(sm)$;
end
Return $dcName$
```

5.1.3. Load aware over cost brokering

The load-aware over cost algorithm maintains a separate table for data center costs. During simulation, it purses the lowest cost data center list to another list involving sender region. Once the lowest cost data center list is created, it distributes all incoming traffic among the lowest cost data centers. Algorithm 3 shows the steps for the load-aware over cost algorithm.

Algorithm 3. Load-aware over cost algorithm.

```
Input: userBase, regionalList, dcXCostList
Output: dcName
Fetch region of the requested sender;
if $regionalList$ is not $NULL$ then
    $listSize \leftarrow Size(dcXCostList)$;
    if $listSize$ is 1 then
        $dcName \leftarrow regionalList.get(0)$;
    else
        getRandomNumber($listSize$) ;
    end
    $dcName \leftarrow dcXCostList.get(sm)$;
end
Return $dcName$
```

5.2. Load balancing methods and proposed algorithm

We may use static or dynamic methods for load balancing. A high variation of loads is observed in the case of cloud computing. Thus, static algorithms are not suitable for a cloud computing environment. Dynamic load balancing is successfully used in cloud and dynamic algorithms that can handle changing workloads over time. Our proposed State-Based Load Balancing (SBLB) algorithm dynamically assigns tasks to idle hosts. It also assigns tasks to available hosts from a task queue. Following this technique prevents the host to become heavily loaded. On the other hand, a task should not wait for a long time in the queue.

When a user sends a request to the cloud system, the cloud controller accepts the request for further processing. The data center controller, service broker and load balancer search for available resources, and allocate the task to an appropriate VM considering request types. In every virtual resource pool, there is a different level of availability of CPUs, memory resources and network bandwidth. On the other hand, each user request has different requirements and constraints. Thus, finding the optimal solution for a user request is considered a complex, multi-faceted problem.

Load-balancing algorithm balances loads among VMs. Once the broker selects the data center for service deployment, the next VM load balancer distributes the loads among VMs in a performance-aware manner. Fig. 4 shows the general workflow of the load balancer. Our proposed SBLB algorithm retains two different tables based on VM states. When a VM reaches its usage threshold, it is placed in the busy state. However, if it does not reach the usage threshold, it is flagged as in the available state.

The data center controller queues user requests and passes them to the load balancer. The load balancer then returns the available VMs from the state table, based on user requests. After allocating the request to the VM, it updates the table accordingly. If the data center controller does not find any available VMs, the data center controller queues the request and waits for resource availability. When processing is finished in a specific VM, the load balancer reallocates the VM for another task. The data center randomly checks for available resources for waiting requests. Algorithm 4 shows the pseudocode of the proposed algorithm. In this algorithm, vmSl denotes the VM state list, while vmSi denotes the VM state index. A list of available and busy VMs are

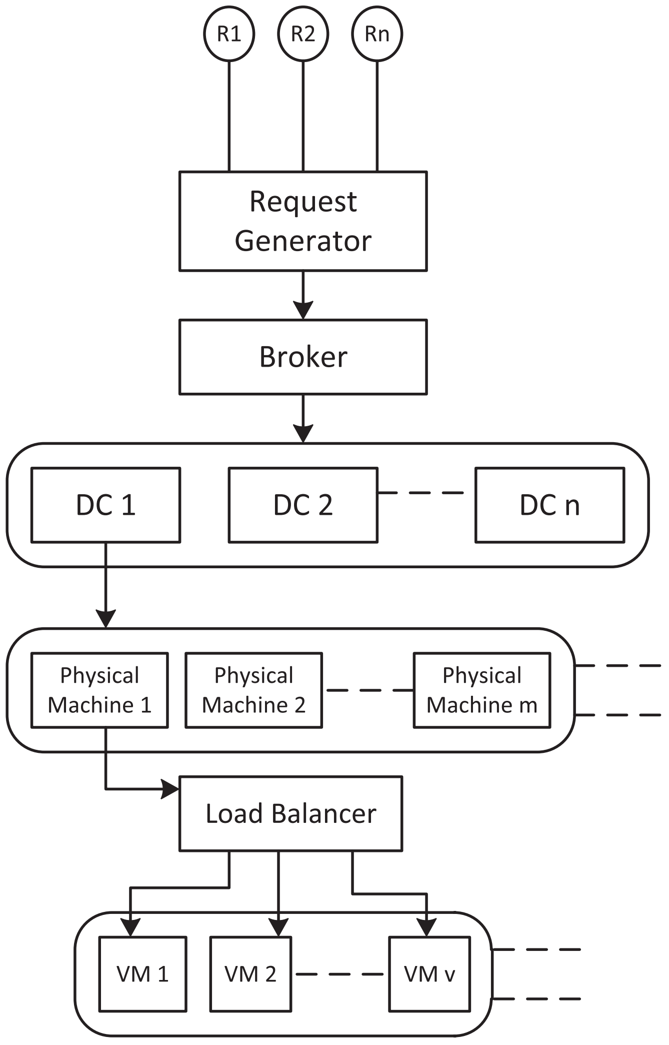

Fig. 4. General flow of VM load balancer.

maintained by vmAv and vmBu respectively.

Algorithm 4. Sate based load balancing algorithm.

```
    Input: vmSl
    Output: vmId
1   if $vmSl$ is $NULL$ then
2       $vmSi \leftarrow getVmSl()$;
3   else
4       if $(vmAv$ is $NULL)$ or $(vmBu$ is $NULL)$ then
5           for all $vmSI$ do
6               if $i$ is $Busy$ then
7                   Add in $vmBu$;
8               else
9                   Add in vmAv
10              end
11          end
12      else
13          $vmId \leftarrow getNextAvailableVm()$;
14          return vmId
15      end
16  end
17  if $e.getId \leftarrow allocated$ then
18      vmBu.put$(vmId, virtualMachineState.BUSY)$;
19  else if $e.getId \leftarrow finished$ then
20      vmAv.put$(vmId, virtualMachineState.AVAILABLE)$;
21  end
```

From the statistics on 30 September 2013 (Facebook User Statistics, 2014), Facebook had 1.19 billion monthly active users worldwide. About 21.85% of the total users worldwide are from North America. We assumed that each active user generates a new request every two minutes. The data center controller is responsible for maintaining workload information with the broker. The peak hour of this region is 15:00–17:00 GMT.

We exhibited a total of seven data centers located in two different regions. One region is where a user base is located. The location for the other data centers was randomly chosen. Every simulated data center consists of five physical machines and a variable number of virtual machines. We have designed six different scenarios by varying the number of VMs up to 15.

The physical machine and the VM configuration of the data center is stated in Table 1. VMs are initially created depending on simulation scenarios. The users and requests are grouped by a factor of 1000 and 100 respectively. In some cases, processing speed was equally divided within VMs based on simulation scenarios. The executable instruction length per request is 250 bytes. The hourly cost per VM in every hour and cost for a 1 GB data transfer are US$ 0.10, according to the old pricing scheme of Amazon EC2. The bandwidth and latency matrix are generated using a Poisson distribution, and only mean values are used. The same technique was used to produce the user base size and interval between user requests. Table 2 shows DC names, along with location and cost per VM, for these data centers.

6. Simulation setup and scenario

We used CloudAnalyst (Wickremasinghe et al., 2010) to conduct our simulation experiments. CloudAnalyst is the extension of the CloudSim (Calheiros et al., 2011) toolkit. CloudSim is a modern simulation framework for Cloud computing which supports large-scale modeling in federated clouds. CloudSim supports large-scale cloud simulations with little or no overhead regarding memory consumption and time initialization. For user requests generation, we used the latest Facebook user statistics throughout the world.

Table 1
Machine configurations.

DC Physical machines configuration	
CPU speed	10,000 MIPS
CPU core	4
RAM	20 GB
Storage	1000 GB
Network bandwidth	10 Gbps
CPU architecture	x86
OS	Linux
Virtual machine monitor (VMM)	Xen
VM configuration	
CPU speed	1000 MIPS
RAM	1 GB
Storage	50 GB
Network bandwidth	100 Mbps

7. Experimental results and discussion

We have simulated our proposed algorithms in a large-scale cloud environment. The processing time is always lower for the

Table 2
Data center location and cost.

DC name	Region name	Region number	Cost per VM (US$/H)
DC1	North America	0	0.30
DC2	North America	0	0.38
DC3	North America	0	0.30
DC4	South America	1	0.15
DC5	South America	1	0.30
DC6	South America	1	0.15
DC7	South America	1	0.73

Load Aware (LA) algorithm. The Cost Aware (CA) algorithm saves cost, but requires more processing time. For all scenarios, the CA always saves costs compared to the LA. The experimental results suggest that the CA is cost effective; however, the LA ensures fastest processing.

Compared to the Service Proximity Service Broker (SPSB) proposed by Wickremasinghe et al. (2010) which was revisited by Naha and Brokering (2014), we found that LA produced a lower average response time with a lower number of VMs; whereas the total cost becomes higher. On the other hand, LA reduced the average processing time with a higher number of VMs in the scenarios; while CA always produced higher processing and response times. Additionally, we developed a Load Aware Over Cost (LAOC) algorithm which improves processing and response times, while ensuring cost effectiveness.

Our research is limited to a brokering mechanism and SLA-aware match making algorithm without altering the existing architecture of cloud providers. One of the limitations of the simulator is that it does not incorporate Failure handling. User traffic data was generated with the assumption of user request generation at five minute intervals. However, in the real world, user traffic is generated in a more complex manner. A detailed analysis of the results is presented as follows.

7.1. Simulation results of cloud brokering algorithms

Our proposed cost aware algorithm is more cost effective compared to the previous algorithm proposed by Wickremasinghe et al. (2010). Performance regarding data center processing and response times decreases; while it also saves costs. Next, we developed another load aware algorithm which upturns performance. In order to make our algorithm more cost efficient, we developed another algorithm with the combination of both the cost and the load-aware techniques. Using the combination of both techniques, we developed a load aware over cost algorithm which is cost effective and performance efficient. The yielded results from the three proposed algorithms are described in the following sections.

7.1.1. Total VM cost of proposed brokering algorithms

On average, our proposed cost aware brokering algorithm is 5.5% more cost effective compared to other previous work. The Service Proximity Based Routing (SPBR-RR) brokering algorithm was proposed previously. Here, the Cost Aware (CA), Load Aware (LA) and Load Aware Over Cost (LAOC) brokering algorithms are the newly proposed algorithms in this work. Fig. 5 shows the comparison of cost between the SPBR-RR, CA-RR, LA-RR and LAOC-RR algorithms. When we focused on load, the total virtual machine cost was also increased. Compared with the CA-RR algorithm, overall costs were increased by 11% with regards to the LA-RR algorithm. Although our proposed load aware algorithm improves DC processing time, cost effectiveness and response time declined.

Furthermore, we developed the LAOC algorithm to eliminate these issues inherent in the mentioned algorithms. We found that

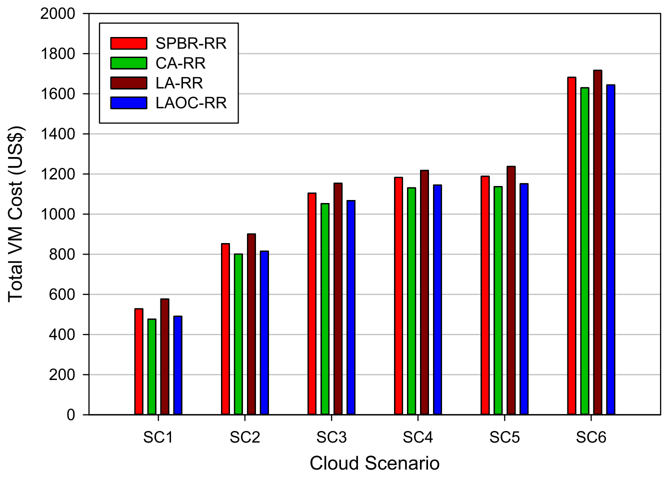

Fig. 5. Total virtual machine cost comparison of proposed brokering algorithms.

total VM costs increase with the proposed LA algorithm. But the LAOC brokering algorithm shows significant improvement with regards to costs. The total VM cost is decreased by 8% compared to the LA brokering algorithm.

7.1.2. Processing time and response time of proposed brokering algorithms

We measured the response and DC processing times for the old and the newly proposed algorithms. Fig. 6(a) and (b) shows the comparison of average and minimum response times. Fig. 6(c) and (d) shows the average and maximum DC processing times. Based on Fig. 6(a), it is clear that average response time has greatly increased. For scenarios 1 and 2, average response time has remarkably increased; this means that response time is higher with a lower number of VMs. Comparing between load aware and cost aware algorithms, overall response time is decreased for scenarios 1 and 2.

However, the overall response time is raised for the remaining four simulation scenarios. We found a 27% improvement for scenarios 1 and 2. On the other hand, a 37% demotion was seen regarding overall response time for the other four scenarios. For LAOC, there is a clear trend of decreasing average response time. The simulation with cloud scenarios 3, 4, 5 and 6 using our proposed LAOC algorithm showed the most surprising degradation of average response time.

On the other hand, the simulation with cloud scenarios 1 and 2 show less improvement compared to other simulation scenarios. Nevertheless, we found a 43% average response time improvement for all scenarios compared to the LB algorithm.

Fig. 6(b) illustrates the comparison between two algorithms in terms of minimum response time. As illustrated in the figure, the minimum response time was slightly increased in our proposed brokering algorithm. However, for some scenarios, especially for 1, 4 and 6, there was no significant difference in minimum response time for both the SPBR-RR and CA-RR algorithms. Our proposed LA algorithm upraises minimum response time for all six cloud scenarios. Based on the results obtained from the simulation of the cost and load aware algorithms, the minimum response time increased by 14% compared to the cost aware algorithm. The proposed LAOC algorithm originates a 13% improvement on minimum response time on average in a comparison between the LA and LAOC algorithms.

As shown in Fig. 6(c), average DC processing time of the proposed CA-RR algorithms is significantly increased compared to the SPBR-RR algorithms. In order to increase performance, we design a new load aware algorithm which increased performance

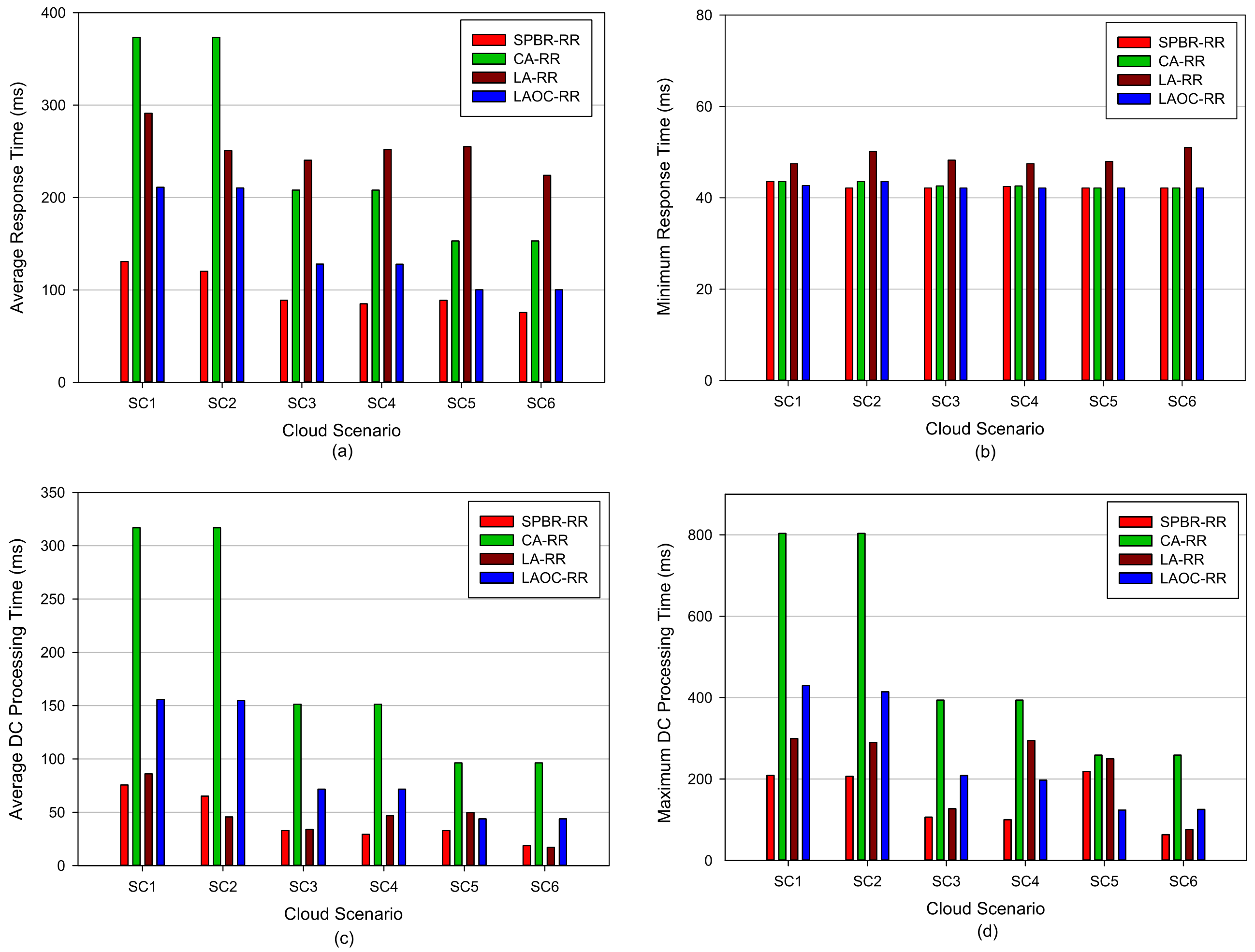

Fig. 6. Response time and processing time comparison of proposed brokering algorithms.

compared with the cost aware algorithm. According to Fig. 6(c), the load aware algorithm reduced average DC processing time compared to the cost aware algorithm. A reduction of 73% was observed on average for DC processing time. Compared with LA-RR, average DC processing time of LAOC-RR was decreased for scenario 5 only, which indicates a 6% improvement. For the other scenarios, the average DC processing time of LAOC-RR increased significantly. For the LAOC-RR algorithm, the average DC processing time was surprisingly increased for cloud scenarios 1 and 2.

Fig. 6(d) shows the difference of the Maximum DC processing time between the old and the newly proposed algorithms. According to the figure, the SPBR-RR average DC processing time was increased for the proposed CA-RR algorithm in all cloud scenarios. Overall, these results indicate that the performance is degraded with respect to response and processing times. The Maximum DC processing time decreased in our proposed load aware algorithm. Compared with the CA-RR, the overall 49% reduction in maximum DC processing time is caused by our proposed load aware algorithm. However, maximum DC processing time is very close to that in the case of scenario 5. Between LA-RR and LAOC-RR, the Maximum DC processing time decreased for cloud scenarios 4 and 5. Moreover, the maximum DC processing time increased. For these two scenarios, the maximum DC processing time significantly decreased by 112% on an average. However, the maximum DC processing time expressively increased in the simulation with other scenarios.

7.2. *Load balancing algorithms*

We measured the total VM cost, average response time, minimum response time, average DC processing time and maximum DC processing time. Our proposed SBLB algorithm outperformed the round robin algorithm. While improving response and processing times, our proposed algorithm attempted to retain existing VM total costs. Thus, using the proposed algorithm conserves the effect on operational costs. Fig. 7 shows a cost comparison with the former and the proposed algorithm.

Our proposed SBLB load balancing algorithm improves average response and DC processing times. In some cases, the minimum response time increased compared with the round robin algorithm. But, overall, the minimum response time improved in our proposed load balancing algorithm. The average response and DC processing times are greatly reduced in the proposed algorithm. The minimum response time and maximum DC processing time were also slightly improved in the proposed algorithm. A detailed description of the experimental results is presented in the following sub-sections.

7.2.1. *Average response time*

The proposed SBLB load balancing algorithm minimizes the average response time for all algorithm combinations. A 23% average response time is minimized using the proposed algorithm. The SBLB load balancing algorithm, with the cost aware brokering algorithm, reduced the average response time by nearly 37%, as shown in Fig. 8(a). The load aware brokering algorithm minimized the average response time by 8%, and the simulation conducted by the combination with the SPBR brokering policy reduced the average response time by 18%. The LAOC algorithm reduced the average response time by 29%. We have simulated the proposed algorithm with SPBR, LA, CA and LAOC cloud brokering algorithms. The simulation results indicate that our proposed SBLB algorithm improves the average response time compared with the round robin load balancing algorithm.

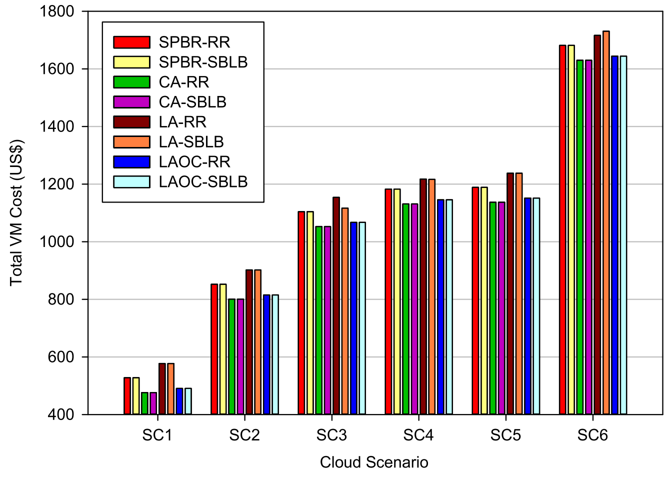

Fig. 7. Total virtual machine cost comparison of proposed load balancing algorithm.

7.2.2. Minimum response time

We found a 1% improvement for the minimum response time with the SBLB algorithm. Some algorithm combinations produced similar or higher response times. For scenarios 3, 5 and 6, the SPBR-RR and SPBR-SBLB algorithms show similar minimum response times, as shown in Fig. 8(b). With the same algorithm combination, the minimum response time was slightly increased in scenario 2. The minimum response time was decreased in scenarios 1 and 4. On the other hand, the CA-RR and CA-SBLB algorithms produced similar results for scenarios 5 and 6. The minimum response time was decreased for the other four simulation scenarios. These four scenarios show a 1.8% improvement over the round robin algorithm. Similarly, the LAOC-RR and LAOC-SBLB algorithms revealed an equal minimum response time for scenarios 3, 4, 5 and 6. However, scenarios 1 and 2 produced a lower minimum response time compared to the round robin algorithm. The simulation results for the LA-RR and LA-SBLB algorithms demonstrated an increase in minimum response time for scenarios 1 and 5. However, minimum response time was decreased for the remaining four scenarios. An average 4.7% improvement was noticed for these four scenarios. Based on Fig. 8(b), it can be concluded that, with the exception of occasional circumstances, the minimum response time was improved by employing the proposed load balancing algorithm.

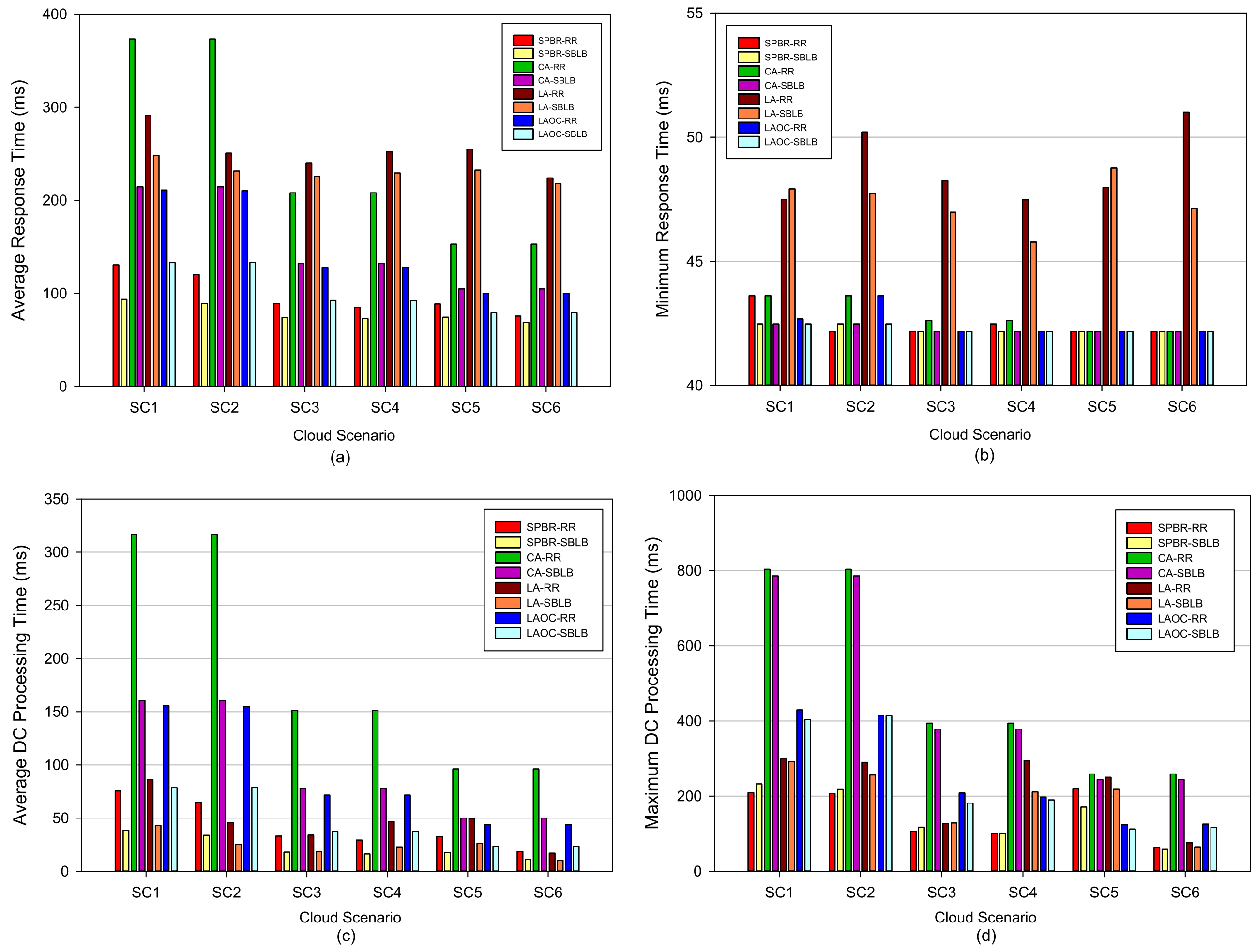

Fig. 8. Response time and processing time comparison of proposed load balancing algorithm.

7.2.3. *Average DC processing time*

The results, as shown in Fig. 8(c), indicate a significant improvement on average in DC processing time. There is a 47% improvement found for average DC processing time compared with the round robin algorithm. Moreover, the SPBR-RR and SPBR-SBLB algorithms show a 46% improvement for DC processing time. However, the improvement is 41% for scenario 6, while similar improvement was witnessed for the remaining five scenarios. Similarly, the CA-RR and CA-SBLB algorithms produced very similar results. A 49% reduction on average for DC processing time was found for the CA-SBLB algorithm compared to the CA-RR algorithm. The simulation results of the LA-RR and LA-SBLB algorithms incurred a 39–51% DC processing time improvement for all scenarios. Overall, a 46% reduction in DC processing time was noted for the LA-RR and LA-SBLB algorithms. On the other hand, the LAOC-RR and LAOC-SBLB algorithms also reduced the similar average DC processing time. Precisely, the average 48% DC processing time was reduced in our proposed load balancing algorithm. Overall, these results indicate that the proposed SBLB load balancing algorithm ensures the fastest task processing in the cloud data center.

7.2.4. *Maximum DC processing time*

Except for the combination of the SPBR-RR and SPBR-SBLB algorithms, all other algorithm combinations produced a lower maximum DC processing time. Fig. 8(d) presents the maximum DC processing time results obtained from the simulations using various algorithm combinations. As shown in Fig. 8(d), the LA-RR and LA-SBLB group reported a significantly less maximum DC processing time compared to the other three groups. On average, the maximum DC processing time was decreased by 11%. The SPBR-RR and SPBR-SBLB algorithm combination improves maximum DC processing time for scenarios 5 and 6. However, the maximum DC processing time increased for scenarios 1–4.

The simulation results for maximum DC processing time of the CA-RR and CASBLB algorithms is depicted in Fig. 8(d). According to the figure, an average 4% improvement can be seen for the maximum DC processing time. The LAOC-RR and LAOC-SBLB algorithms produced a 7% maximum DC processing for time advancement. The highest decrement of maximum DC processing time is observed for cloud scenario 3, which is 13%; while the lowest decrement in maximum DC processing time is observed for cloud scenario 2, which is 0.24%. In summary, these results show that our proposed load balancing algorithm reduced the maximum DC processing time.

8. Significant contributions

The main contributions in this study are twofold. The first contribution involves the three proposed cloud brokering algorithms, referred to as CA, LA and LAOC. The second contribution involves the load balancing algorithm, which is known as the SBLB algorithm. The CA brokering algorithm saves 5.5% in costs compared to the existing one; but response time and the DC processing time are highly increased. On the other hand, the LA algorithm shows a 37% improvement in average response time with a higher number of VMs in the scenarios; and a 73% improvement in average DC processing time. However, the LA algorithm incurred 11% extra costs compared to the CA algorithm. The proposed LAOC algorithms is a midway solution based on the CA and LA algorithms. The LAOC incurred higher costs from the CA and lower costs from the LA brokering algorithm. Similar trends were found for response and processing times. The proposed SBLB load balancing algorithm improved DC processing time and response time by retaining same total VM costs. Therefore, it can be summarized that the proposed SBLB algorithm improved the DC processing and response time without affecting the total VM costs. Moreover, the proposed brokering algorithms help cloud users to choose the appropriate providers based on their cost saving or performance aware constraints. The proposed algorithm helps service providers to manage their resources.

9. Conclusion

The demand for cloud computing has greatly increased in the past few years due to the advancement in computing as a service form. In the cloud computing concept, users are able to utilize computing resources according to their needs and requirements. The cloud approach helps users to reduce the cost of IT infrastructures. To provide services, different cloud service providers build their own computing platforms differently, due to the lack of a common standard. Selecting the cloud provider from among these heterogeneous cloud environments is a challenging task for users. A broker is capable of finding an appropriate service provider that would satisfy user service requirements in terms of a Service Level Agreement. Load balancing in the cloud is another important research issue. Cloud Computing offers on-demand provisioning of computing resources to users. Cloud service providers manage a large number of user requests to provide services according to user demands. Allocating and managing user requests to physical hardware is a challenging issue, since there is a need to create a load balance among available system resources. Effective load balancing saves operational costs, improves user satisfaction and leads to accelerate overall performance.

The results of the simulations carried out show that cost saving brokering requires a greater processing time, and sometimes, a greater response time. However, the fastest processing brokering leads to higher utilization costs. Depending on user requirements, the broker decides whether the process is performed in a cost saving manner, or a faster processing technique. The results of this study indicate that an efficient brokering technique could save time and cost for cloud users. The simulation results show that the performance of the SBLB algorithm is greatly improved when compared with the existing algorithms in a simulated environment. The findings of this study suggest that an efficient load balancing algorithm could minimize execution time, which would be of benefit to cloud users and providers. The broker should have the capability to handle changing user requests and dynamic resource allocation based on SLA. The aim to resolve these issues is regarded as future work. Another important limitation lies in the fact that we have conducted our experiments in a simulated environment. Further investigation and experimentation in real-world cloud brokering and load balancing mechanisms is strongly recommended.

Acknowledgment

The authors would like to thank the Malaysian Ministry of Education for provide funding under a Commonwealth Scholarship and Fellowship Plan (CSFP) scholarship. This work is also partially supported by the Malaysian Ministry of Education under the Fundamental Research Grant Scheme FRGS/02/01/12/1143/FR.

References

Akhter, N., Othman, M., 2014. Energy Efficient Virtual Machine Provisioning in Cloud Data Centers. In: International Symposium on Telecommunication Technologies (ISTT), IEEE, Langkawi, Malaysia, pp. 282–286.

Akhter, N., Othman, M., 2016. Energy aware resource allocation of cloud data center: review and open issues. Clust. Comput., 1–20.
Bernstein, D., Vij, D., Diamond, S., 2011. An intercloud cloud computing economy-technology, governance, and market blueprints. In: 2011 Annual SRII Global Conference (SRII), IEEE, San Jose, CA, USA, pp. 293–299.
Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S., Morrow, M., 2009. Blueprint for the intercloud-protocols and formats for cloud computing interoperability. In: 2009 ICIW'09 Fourth International Conference on Internet and Web Applications and Services, IEEE, Mestre, Italy, pp. 328–336.
Bernstein, D., Vij, D., 2010. Intercloud directory and exchange protocol detail using XMPP and RDF. In: 6th World Congress on Services (SERVICES-1), IEEE, Miami, Florida, USA, pp. 431–438.
Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I., 2009. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. 25, 599–616.
Calheiros, R.N., Toosi, A.N., Vecchiola, C., Buyya, R., 2012. A coordinator for scaling elastic applications across multiple clouds. Future Gener. Comput. Syst. 28, 1350–1362.
Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R., 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw.: Pract. Exp. 41, 23–50.
Domanal, S.G., Reddy, G.R.M., 2014. Optimal load balancing in cloud computing by efficient utilization of virtual machines. In: 2014 Sixth International Conference on Communication Systems and Networks (COMSNETS), pp. 1–4.
Facebook User Statistics, 2014. ⟨http://newsroom.fb.com/Key-Facts⟩(accessed 06.12.13).
Fang, Y., Wang, F., Ge, J., 2010. A task scheduling algorithm based on load balancing in cloud computing. In: Web Information Systems and Mining, Springer, Sanya, China, pp. 271–277.
Ferreto, T.C., Netto, M.A., Calheiros, R.N., De Rose, C.A., 2011. Server consolidation with migration control for virtualized data centers. Future Gener. Comput. Syst. 27, 1027–1034.
Florence, A.P., Shanthi, V., 2014. A load balancing model using firefly algorithm in cloud computing. J. Comput. Sci. 10, 1156–1165.
Garg, S.K., Toosi, A.N., Gopalaiyengar, S.K., Buyya, R., 2014. SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter. J. Netw. Comput. Appl. 45, 108–120.
Hu, J., Gu, J., Sun, G., Zhao, T., 2010. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In: 2010 Third International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), pp. 89–96.
Jrad, F., Tao, J., Streit, A., 2012. Simulation-based evaluation of an intercloud service broker. In: The Third International Conference on Cloud Computing, GRIDs, and Virtualization, Cloud Computing 2012, pp. 140–145.
Kessaci, Y., Melab, N., Talbi, E-G., 2013. A Pareto-based genetic algorithm for optimized assignment of vm requests on a cloud brokering environment. In: 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 2496–2503.
Krishna, P.V., 2013. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl. Soft Comput. 13, 2292–2303.
Lucas-Simarro, J.L., Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M., 2013. Scheduling strategies for optimal service deployment across multiple clouds. Future Gener. Comput. Syst. 29, 1431–1441.
Maguluri, ST., Srikant, R., Ying, L., 2012. Stochastic models of load balancing and scheduling in cloud computing clusters. In: 2012 Proceedings IEEE INFOCOM, pp. 702–710.
Mahajan, K., Makroo, A., Dahiya, D., 2013. Round robin with server affinity: a VM load balancing algorithm for cloud based infrastructure. J. Inf. Process. Syst. 9, 379–394.
Naha, R.K., Othman, M., 2014. Brokering and load-balancing mechanism in the cloud-revisited. IETE Tech. Rev. 31, 271–276.
Naha, RK., Othman, M., 2014. Optimized Load Balancing for Efficient Resource Provisioning in the Cloud. In: International Symposium on Telecommunication Technologies (ISTT), IEEE, Langkawi, Malaysia, pp. 382–285.
Naha, R.K., Othman, M., Akhter, N., 2015. Evaluation of cloud brokering algorithms in cloud based data center. Far East J. Electron. Commun. 15, 85–98.
Quarati, A., Clematis, A., Galizia, A., DAgostino, D., 2013. Hybrid clouds brokering: business opportunities, QoS and energy-saving issues. Simul. Model. Pract. Theory 39, 121–134.
Randles, M., Lamb, D., Taleb-Bendiab, A., 2010. A comparative study into distributed load balancing algorithms for cloud computing. In: 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 551–556.
Rochwerger, B., Breitgand, D., Epstein, A., Hadas, D., Loy, I., Nagin, K., et al., 2011. Reservoir—when one cloud is not enough. IEEE Comput. 44, 44–51.
Sun, L., Dong, H., Hussain, F.K., Hussain, O.K., Chang, E., 2014. Cloud service selection: state-of-the-art and future research directions. J. Netw. Comput. Appl. 45, 134–150.
Tordsson, J., Montero, R.S., Moreno-Vozmediano, R., Llorente, I.M., 2012. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future Gener. Comput. Syst. 28, 358–367.
Van den Bossche, R., Vanmechelen, K., Broeckhove, J., 2010. Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. In: 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), pp. 228–235.
Vecchiola, C., Chu, X., Mattess, M., Buyya R, 2011. Aneka—integration of private and public clouds. In: Cloud Computing Principles and Paradigms, Wiley, Hoboken, pp. 251–274.
Wang, S-C., Yan, K-Q., Liao, W-P., Wang, S-S., 2010. Towards a load balancing in a three-level cloud computing network. In: 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol. 1, pp. 108–113.
Wickremasinghe, B., Calheiros, RN., Buyya, R., 2010. Cloudanalyst: a cloudsim-based visual modeller for analysing cloud computing environments and applications. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 446–452.

Ranesh Kumar Naha awarded Master of Science from the Faculty of Computer Science and Information Technology in Universiti Putra Malaysia in 2015. He received B.Sc. degree in Computer Science and Engineering from State University of Bangladesh in 2008. His research interests are in wired and wireless network, parallel and distributed computing and cloud computing. During his master study he was awarded Commonwealth Scholarship provided by Ministry of Higher Education, Malaysia. After his graduation he served as Lecturer until 2011 in Daffodil Institute of IT, Bangladesh.

Mohamed Othman received his Ph.D. degree from the Universiti Kebangsaan Malaysia with distinction (Best Ph.D. Thesis in 2000 awarded by Sime Darby Malaysia and Malaysian Mathematical Science Society) and now he is a Professor in Computer Science at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). He served at various positions in UPM, member of Computer Chapter Malaysia Section and as chief-editor and associate editor of several computer and network journals. He is also an associate researcher at the Computational Science and Mathematical Physics Lab., Institute of Mathematical Research (INSPEM), UPM. His main research interests are in the fields of parallel and distributed computing architecture, interconnection ultra speed network, network design and management (wireless communication and network, and mobile traffic monitoring), and scientific computing. He is a member of IEEE Computer Society, International Association of Computer Science and Information Technology, IAENG, Malaysian National Computer Confederation and Malaysian Mathematical Society. He already published more than 500 National and International journals and proceeding papers. His main research interests are in the fields of parallel and distributed algorithms, high performance computing architecture, wireless communication and network, network design architecture and management (high-speed, security, wireless, and traffic monitoring) and scientific computing.