Multimed Tools Appl (2017) 76:14485–14509
DOI 10.1007/s11042-016-3852-x

Quality of service aware cloud resource provisioning for social multimedia services and applications

Tamal Adhikary[1] · Amit Kumar Das[1] · Md. Abdur Razzaque[1] · Majed Alrubaian[2] · Mohammad Mehedi Hassan[2] · Atif Alamri[2]

Received: 29 February 2016 / Revised: 5 July 2016 / Accepted: 9 August 2016 /
Published online: 22 August 2016

Abstract The increasing number of next-generation multimedia services and social media applications in cloud computing put additional challenges in efficient resource provisioning that targets to minimize under or over utilization of resources as well as to increase user satisfaction. Most of the works in the literature focused either on resource estimation and scheduling approaches or energy consumption for executing social media data processing applications. However, they do not consider energy consumption cost for communication devices and network appliances and schedule Virtual Machines (VMs) based on centralized job placement approach. In this paper, we develop a Quality of Service (QoS) aware cloud resource management system that decreases energy consumption and increases resource utilization by diverse multimedia social applications. In order to minimize the VM creation time we allow recycling of VM resources for user request with similar resource requirements. We have developed two distributed and localized resource

✉ Md. Abdur Razzaque
razzaque@du.ac.bd

Tamal Adhikary
tamal.csedu@gmail.com

Amit Kumar Das
amit.csedu@gmail.com

Majed Alrubaian
malrubaian.c@ksu.edu.sa

Mohammad Mehedi Hassan
mmhassan@ksu.edu.sa

Atif Alamri
atif@ksu.edu.sa

[1] Green Networking Research Group, Department of Computer Science and Engineering, University of Dhaka, Dhaka, Bangladesh

[2] Research Chair of Pervasive and Mobile Computing, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia

management algorithms based on energy conservation, and requirements and availability of resources. The results of simulation experiments depict that the resource scheduling system greatly reduces the amount of energy consumption while maintaining the QoS of social multimedia applications.

Keywords Multimedia healthcare applications · Ubiquitous social media computing · Resource provisioning · Collaborative resource utilization · Quality of service

1 Introduction

Cloud computing has become an indispensable technology trend of modern highly evolving world. It has widely attracted the attention of governments, industries and organizations of all kinds due to the ease of implementing and managing distributed computing environment. New era of proving massive processing and storage service has been commenced with the commercial deployment of this computing paradigm. However, little attention has been given to the convergence of cloud computing and social multimedia applications [13, 37] to handle a huge amount of media social collaboration with respect to various social variables of clients or users (for example, client profile, feeling or client suppositions, social connection among the clients, and so forth) and sharing social media content such as video, audio etc. Nevertheless, the increasing number of social media applications in cloud can put additional challenges in efficient resource provisioning that targets to minimize under or over utilization of cloud resources as well as to increase user satisfaction [18].

The lone concern for the deployment of Cloud Data Centers (CDCs) has been attributed to the high performance gain omitting the apprehension of energy consumption. However, on an average, a CDC consumes 25,000 households energy [20] and thus energy-efficient cloud resource scheduling has become an utmost important, giving birth the concept of Green Cloud Computing (GCC). The vision of GCC is to integrate computation and communication resource management in such a way that, energy-efficiency can be ensured along with robustness and quality of service (QoS). The GCC aims to gain multiple conflicting objectives - QoS in service level agreement (SLA) and minimization of energy consumption.

Energy aware VM scheduling for dynamic load balancing can reduce energy consumption, but much more consideration is necessary for holistically examining where and how such mechanisms are needed [26]. Again, providers need to be concerned about energy-efficiency while VM provisioning, that is while reusing, renting, generating new VMs or terminating one which task execution time exceeds the remaining billing time unit (BTU). A dynamic, efficient, adaptive and automated VM scheduling and VM provisioning methodology can greatly reduce the energy consumption in a CDC.

The existing solutions in the literature addressing the aforementioned challenges can be classified broadly into two major categories. The first category of the solutions focuses on maintaining the SLA through QoS aware resource allocation to user requests [6, 9, 15, 22, 38]. However, under or over provisioning of resources, while maintaining QoS strictly, would increase the service as well as the energy costs. The second category of solutions aims to reduce energy consumption through energy aware VM distribution in the CDC [1, 3, 4, 21, 36]. However, VM distribution among CDCs increases communication latency, request service time and degrades the QoS level.

In this work, a resource management framework, namely VSA, has been developed for social multimedia applications that achieves energy-aware adaptive VM scheduling and

provisioning while maintaining the required level of QoS. The VSA system provides automated, flexible and dynamic resource management in a multi-cluster CDC. We have also analyzed the impacts of energy-aware VM scheduling on CDC infrastructure. The model achieves target QoS in terms of response time by controlling the admittance of the requests. The requests admittance are controlled in such a way that, accepted requests do not experience a delay grated than the time limit specified in the SLA. Service time of a request is reduced by recycling VMs and thus QoS is met by decreasing the response time. Requests are categorized into multiple classes based on similarity in demand of cloud resources and a input queue is created for all the requests belonging to the same class. Once VMs are created to provide services to a particular class of requests, the same VMs can be recycled for the other requests belonging to the same request class. Policies have been developed to create a new VM following the priorities of the requests and the resources available. A conference version of this work has been published in [9], where we only consider QoS aware VM provisioning method for single-cluster CDC.

The key contributions of the proposed VSA framework can be summarized as follows:

- A new cluster architecture and a multi-cluster cloud infrastructure design in VSA forms a ubiquitous multimedia computing platform that reduces the management complexity of the CDC resources.
- Multi-level priority queues, controlled request admittance and VM reusing approaches of VSA ascertain user QoS.
- Energy aware intra- and inter-cluster VM scheduling algorithms have been developed to enhance energy conservation of the total system.
- The simulation performance results, carried out in CloudSim [7], show that the proposed VSA system achieves as much as 50 % improvements in terms of QoS and 45 % improvements in terms of energy respectively, compared to the state-of-the-art works.

The remaining parts of this paper is organized as follows. The related works and motivations of this work are presented in Section 2. The Section 3 presents a new cluster architecture and a multi-cluster cloud computation environment that describe system model of our work. The Section 4 illustrates the proposed QoS aware VM provisioning algorithms and in Section 5, energy-efficient inter- and intra-cluster VM scheduling algorithms are presented. The simulation environment, performance metrics and the results of experiments, carried out in CloudSim [7], are discussed in Section 6. In Section 7, we conclude the paper and state a few directions for future works.

2 Related works

As the smart devices are getting popular very rapidly, the volume of ubiquitous social multimedia contents from these devices is increasing exponentially. Social multimedia applications produce and consume contents of different types like 2-D/3-D videos, images, audio clips, documents of various formats, music clips and the processing, storing, distribution and security issues of these contents require higher computing facilities provided by the cloud data centers (CDCs) [17, 19]. A number of recent studies have signified the necessity, probable architecture and management of the multimedia contents [25, 35]. Recent studies have shown that, deployment of social multimedia applications in CDCs is increasing rapidly now-a-days [14]. The core reason behind the scene is the infrastructure and maintenance cost of the bulky amount of resources which the CDCs can easily provide. Cloud

providers (CPs) also maintain large and power-consuming CDCs to provide a certain level of elasticity and scalability. Therefore, the problem of energy consumption at the client edge can be reduced with the cost of huge amount of energy consumption at the server side which was one of the major commercial credentials behind cloud computing [8, 24, 32].

In order to guarantee SLA, over allocation of resources has been proposed in [27], which is based on overbooking theory [31]. Over allocation of resources confirm that the actual resource need to support the service would meet up. However, this approach fails to achieve optimized utilization of resources.

Live VM migration schemes, Wake-on-LAN (WoL) and Dynamic Voltage and/or Frequency Scaling (DVFS) has been considered in [33] to resize the active server pool and reduce energy consumption from non-active servers. The authors of [5] have proposed an architecture for providing services to multi-tier applications exploiting the mechanisms of queuing networks. The major problem in the architecture is that, the number of VMs are not altered dynamically with the change in computation load.

Local and global policies for the virtualization technologies, being actively deployed into large-scale CDC environments, have been proposed in [28]. Though the system of local policies has been clearly illustrated using the strategies of guest operating system power management, how the global policies maintaining QoS requirement of the customer requests has not been discussed in details.

The authors in [4] categorized resource sharing methodology into local and global policies to establish an energy management system for virtualized CDCs. The analytical results based VM provisioning have been presented in [6], where the authors have developed a dynamic model of VM deployment. However, they didn't analyzed the difference in the resource requirements of incoming user requests. They have created a new VM for every incoming requests even in cases where VMs are available for serving requests with same degree of resource requirement. This approach kills a lot of time since VM creation and deletion are very much time consuming.

Frameworks for QoS aware mobile cloud computing leveraging adaptive QoS management has been proposed in [2, 38]. The authors have used FCM (Fuzzy Cognitive Map) to model the QoS aware VM provisioning system. However, the proposed system does not address a number of problems like the number of accepted requests at a particular time period, the request serving and congestion handling technique etc. In this work, we have clearly specified these issues related to request management and VM provisioning.

The authors of [23] have used SaaS mashup applications to develop a queuing infrastructure that targets to calibrate the optimal number of VM instances for user applications to maximize the benefits of the SaaS providers. In 'Claudia' system [30], elasticity rules and performance indicators are used to provide resources to user requests considering user-defined constraints. Reactive approach has been used in 'Claudia' for achieving QoS, which fails to provide QoS in some cases and the SLA violations result in degradation of CP's reputation. For enhancing QoS in cloud infrastructure, the authors of the paper [10, 12, 16, 34] also propose some new techniques.

Resource provisioning in cloud computing has started employing CDCs and deploying VMs in these CDCs in order to maximize the utilization of the resources [39]. The two key advantages of cloud platforms are efficient sharing and dynamic scaling of the resources in the CDCs. These are achieved through virtualization of the resources. Resource virtualization also allows fault isolation and improved manageability of the CDC resources.

Our proposed framework VSA has similarity with AVM [6]. However, there are a number of distinct differences between them. *Firstly*, our proposed VSA system forms clusters of physical servers in the entire network and thus the user requests are handled more efficient in a distributed manner, opposing to AVM that uses centralized approach. *Secondly*, the VSA categorizes user requests into groups and processes user requests accordingly in allocating same type of VMs, reducing the VM creation and destroying overheads; whereas, the AVM always creates a new VM for serving an incoming user request since it does not make groups of similar requests. *Thirdly*, our proposed VSA system provides a QoS aware VM allocation algorithm based on VM recycling, allowing to handle more user request with QoS compared to AVM since the latter one does not use any VM pools and thus it suffers from resource constraint. *Finaly*, the energy-aware VM scheduling algorithms of VSA system minimizes energy expenditures of the CDCs; whereas, the AVM provides no such mechanisms for checking the energy expenses.

3 Network model and assumptions

For meeting up SLA and allocating resources to the social media application users according to the agreement, a QoS aware CDC modeling is needed. Management of QoS for a whole CDC is a complex task and for leveraging such complexity, a number of servers can be grouped. What follows next are the details architecture of a complete CDC and VM resource requirement prediction methodology for serving the user data processing requests.

3.1 System architecture

The working environment and the assumptions considered in the VSA model are concisely presented in this section. The system consists of a CDC, which is divided into a number of clusters containing physical servers. We consider heterogeneous servers with different capacities that provide a range of computing resources. The VM number and the resource distribution in the VMs are configured dynamically to comply with load variance in the CDCs. Let T_{nneg} is the time specified in SLA which is set through the negotiation between the user and the CDC and T_{rsp} is the system predicted response time that the CDC consumes to serve the request. The condition $T_{rsp} \leq T_{nneg}$ must be achieved to satisfy the QoS requirements.

We also assume that the requests received by the CDCs are grouped into a number of classes considering their requirements. A number of CPs provides services based on request classes. For example, there are 11 types of instances in Amazon EC2. Resource configuration of each VM instance in terms of computing power (processor speed in MIPS), primary and secondary storage, bandwidth and successful I/O operations varies from one another [28]. The notations we have considered in this paper are summarized in Table 1.

In Fig. 1, the design components of a cluster is shown. There is an *Information Record Database (IRDB)* in each cluster in which the record of previous tasks are stored. *Resource predictor* estimates the resource requirement for processing a request. Hence, it estimates the response time that will be required T_{rsp} for completing the service of a request. *Resource manager* checks for available resources and decides whether it will serve a new request or not. *Resource allocator* serves VM to the user for processing user requests. *Resource monitor* continuously observes the resource utilization of the servers and the clusters. Whenever

Table 1 Notations

Notation	Description
Q	Set of all input queues
T_{req}	Time requested by the user
T_{rsp}	Time required to provide response for newly arrived request
T_{neg}	Negotiated time with the users
T_{mean}	Monitored mean execution time
T_{ttl}	Total required time
T_{nneg}	New negotiated time
T_{srv}	Total affordable service time
ϕ_{ru}	Total usage of resource in the cluster
ϕ_{ar}	Total available resources in the cluster
μ	Occupancy ratio
σ_{tu}	Total usage of resources
σ_{ta}	Total available resources
ω	Workload ratio
V_O	Neighborhood Occupancy Vector
T_{com}	Communication time period
E_c	Energy conservation amount
Ei	Energy consumption amount
M_{CC}	Mean computation cost

resource requirement of a user request changes beyond the capacity of the allocated VM, the *resource manager* tries to allocate a new VM to the user request according to its increasing necessity in a self-adaptive manner. User requirement changes are also kept in the *IRDB* for better prediction.

In Fig. 2, we see that user applications from healthcare social media services are executed on cloud virtual machines (VMs) that are hosted on physical servers. The total arrangement of CPs is divided into clusters. Each cluster is equipped with data storage, computation and resource management units. Resource management unit works as a local scheduler for the cluster. It allocates computing resources to user requests. Each cluster maintains communication to all the other clusters adjacent to it. The neighbors share computational loads with their neighbors whenever necessary. Migration of VMs from one server to another or job distribution among the servers is faster and more convenient within a cluster. Whenever a cluster runs out of resources and can't satisfy SLA for requests, it can take help from adjacent neighbors. While distributing jobs to the neighboring clusters, computation load of the clusters is kept in consideration. Service downtime accounted for migrating VMs from one server to another in the same cluster or between two clusters is quite short as migration is preformed on-the-fly using 'demand migration' [29] policy. That is, an identical image of the migrated virtual machine is created in the remote server first. The user gets access to the new VM, when the new VM image is created.

Each cluster and the servers in the clusters operate in one of the two modes: $sleep$ and $active$.

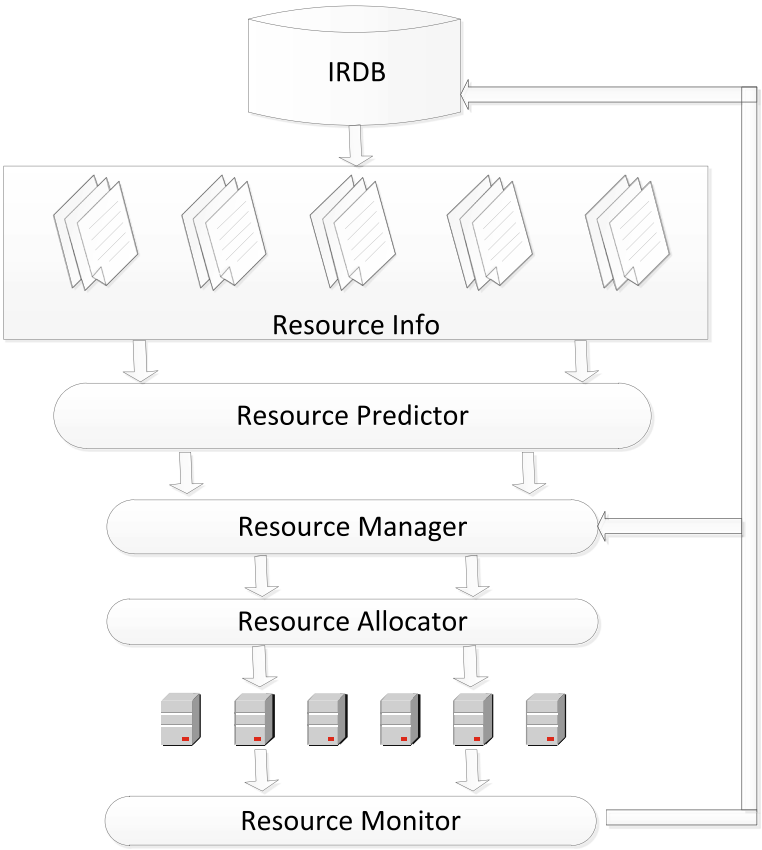


Fig. 1 A single cluster architecture

- *Active:* This is the full functional state of a cluster. The resource manager and a number of active servers process the requests from different users.
- *Sleep:* This is the state where all working machines of a cluster are switched off. Only the communication maintenance servers remain working at that period.

3.2 Prediction of request execution time

This prediction unit is responsible to anticipate the execution time of the current request based on the historical data on previous experiences. An LPF (Low-Pass-Filter) equation is applied for calculating the predicted mean execution time by implementing a degrading function. The LPF maintains a good performance for predicting current behavior when assuming historical data [11]. The LPF methodology can be shown by using the following equation,

$$T_{rsp}^{k} = w \times T_{rsp}^{k-1} + (1 - w) \times V, \tag{1}$$

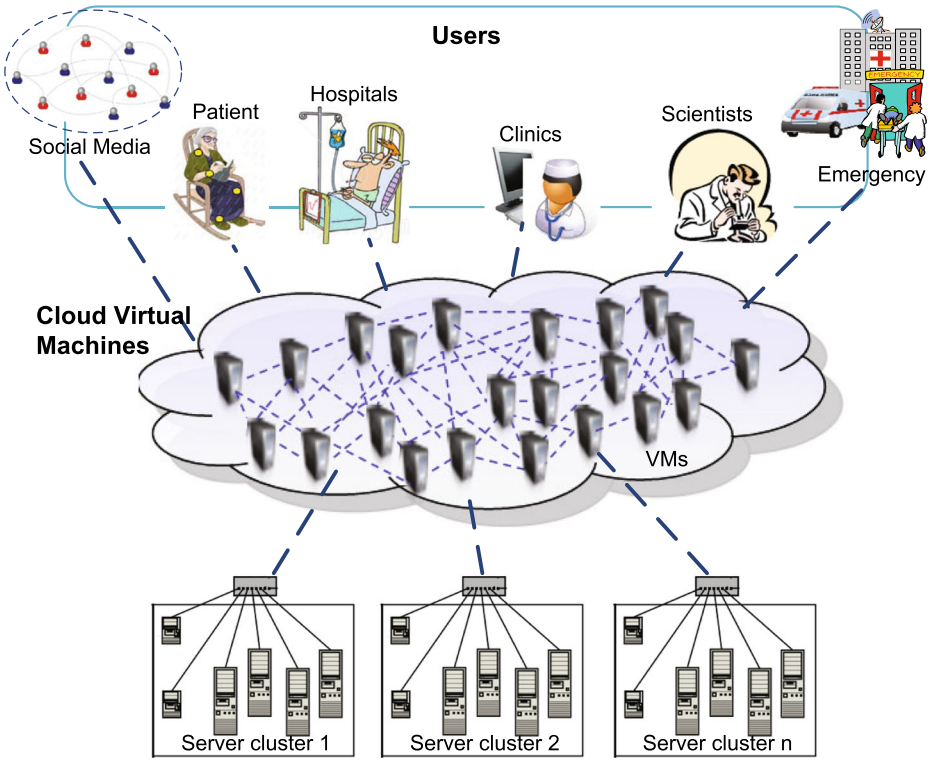


Fig. 2 Multicluster data center environment

where, T_{rsp}^{k} is the predicted level of time requirement to serve a newly arrived user request, w is the constant of degradation, T_{rsp}^{k-1} is the previously predicted value on the same type of requests, V is the actual average execution time required in the previous run. In our approach, we have used this method for predicting working time of the user requests.

Data transmission latency and energy consumption between two server nodes connected directly using the same network switch is very low compared to those servers connected via a hierarchy of switches [3]. The VSA system supports energy conservation and delay minimization by prioritizing the data transfer among the servers connected via the same switch.

4 QoS aware resource provisioning

In this section, cluster components and their enrollment in QoS aware resource provisioning have been proposed. How a clustering system works is described first. Then, a VM provisioning algorithm is developed that ascertains QoS for the user requests. Finally, an in depth analysis on QoS accomplishment is presented.

4.1 Cluster modeling for QoS aware VM provisioning

An adaptive algorithm that aims to satisfy target QoS while providing VMs for serving user applications is presented in this subsection. The algorithm takes into account the dynamic change of workload at the data centers along with the uncertain behavior of the network components. The QoS aware self-adaptive mechanism for VM provisioning is depicted in Fig. 3.

New requests from the user enter the CDCs through the *Admission Controller + Resource Predictor*. If a newly arrived request has the waiting time (t_{wait}) longer than the time needed to create a new VM (t_{VM}), a VM with the resource requirement specified in the request SLA is created and provided to that request. Therefore, t_{wait} must follow the inequality (2),

$$0 \leq t_{wait} \leq t_{VM}. \tag{2}$$

When the CDCs get jam-packed with user requests, guaranteeing QoS for newly admitted requests become impossible. The admission controller does not allow requests to enter the CDCs in such a case since no SLA for the new requests can be promised. *Resource Predictor* determines the amount of computational resource required (as described in Section 3.2) and selects the queue of request class where the new request should be placed.

The *Resource Manager* determines the available resources in the CDCs and dictates the *Admission Controller + Resource Predictor* unit to take decision whether to accept new requests or not. The resource requirement for the requests in each VM and the current status of the servers are monitored continuously by the *Resource Monitor*. The *Resource Allocator*

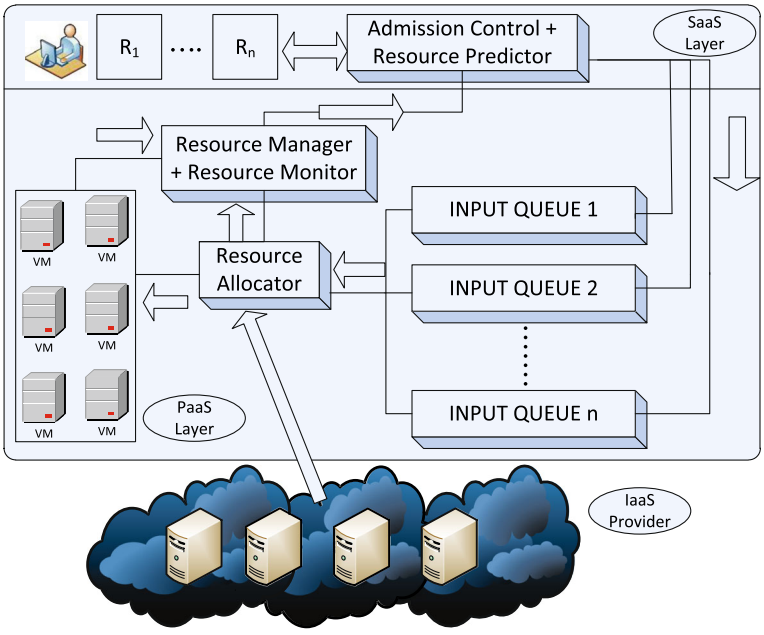


Fig. 3 QoS aware VM provisioning through self adaptation

allocates an existing or a new VM to a user request following its time criticality. While provisioning VMs to requests, *Resource Allocator* uses the parameter P_{fact}^{ij} which describes the priority of request i in queue j, as derived in Algorithm 1.

Figure 4 describes the dynamic queue management process for assisting QoS-aware VM provisioning. The *Admission Controller + Resource Predictor* first determines the free resources and then identifies the resource requirement of the incoming requests. It permits a request to enter the CDCs, if it can guarantee the execution of the requests within the time specified in the SLA. Whenever a new request is let to enter the system, a suitable queue for the request is selected by accessing the resource demand of the request. If the resource demand of the new request does not match with any existing classes, the request is paced in a new queue containing only itself. *Resource Monitor* continuously observes the resource demand of the running requests in the VMs. If the initial prediction of resources for a request is wrong or the requirement changes with time, the VM serving the request can be resized up to a specified threshold level. In cases, when the resizing of VM changes the class of the request, a new VM is provisioned to serve the additional resource demand of the request.

The VSA system do not create new VM every time a request enters the CDCs. Only time critical requests are provided with new VMs. All the other requests can reuse the already created VMs of their same class. *Resource Allocator* keeps track of all the running requests and requests in queue, determines which VM is redundant therefore can be deleted and which type of VM needs to be created for time critical requests. In this way, QoS of the user requests can be ensured by completing all the requests within the negotiated service time.

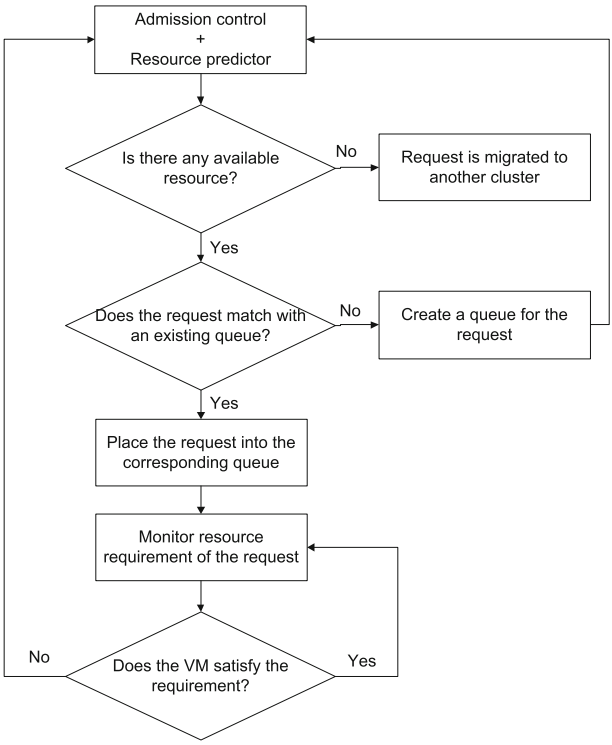


Fig. 4 QoS-aware request-queue management

4.2 Resource provisioning algorithm

Algorithm 1 Resource Provisioning Algorithm

INPUT:
T_{rsp} : *Time required to provide response for newly arrived request to satisfy QOS*,
T_{mean} : *Monitored mean time of execution*,
T_{srv} : *Total affordable time that the CDC will require to server requests*,
$ReservedTime$: *Time reserved to cope with dynamic network and workload behavior*,
n : *Number of requests in service*,
OUTPUT: *A vector* V_{map} *that maps requests to VMs*

1: $T_{neg} \leftarrow \sum_{l=0}^{n} T_{req}^{l}$
2: $T_{ttl} \leftarrow n \times T_{mean}$
3: $T_{estm} \leftarrow T_{rsp} + T_{ttl} + ReservedTime$
4: $T_{max} \leftarrow T_{neg} + T_{nneg}$
5: **if** $T_{max} \geq T_{estm}$ AND $T_{srv} \geq T_{max}$ **then**
6: **if** $T_{rsp} \leq T_{nneg}$ **then**
7: $V_{map} \leftarrow V_{map} \bigcup \{VM, new\_request\}$
8: **end if**
9: **else**
10: Reject job from entering into input queue
11: **end if**
12: Calculate P_{fact} using Eq. 5
13: **if** An already created VM is available for the queue **then**
14: Host request in a VM
15: **else**
16: **for** each queue j in Q **do**
17: $Priority_j \leftarrow \sum_{i=0}^{|j|} P_{fact}^{ij}$ {sum of all request priorities}
18: **end for**
19: $j_{max} \leftarrow \underset{j \in Q}{\text{argmax}}\ Priority_j$
20: $i_{max} \leftarrow \underset{i \in j}{\text{argmax}}\ P_{fact}^{ij}$
21: **if** resources are available for i_{max} **then**
22: Create new VM for i_{max}
23: **else**
24: Wait for VM of same queue type j
25: **end if**
26: **end if**

The VSA system ascertains QoS through controlling the entrance of requests and provisioning VMs judiciously. Whenever a new request tries to enter a CDC, total negotiated time corresponding to the requests in service and in the queue i.e., all the request in the CDC is calculated and it is given by T_{neg}. Thereafter, the completion time for all the requests in the CDC is estimated using the monitored mean execution time T_{mean} of the requests of all request classes. The estimated total required service time is denoted as T_{ttl}. It is then added with a reserved time period determined by the system to guarantee the completion of execution considering the time varying workload and uncertainty in communication delay.

Whenever a new request applies to enter the CDCs, *Admission Controller + Resource Predictor*, estimates T_{rsp} for that request which is the time that will be needed actually to serve the request. It is then allowed to enter the CDC if the following inequalities hold.

$$T_{nneg} + T_{neg} \geq T_{ttl} + T_{rsp} + Reservedtime \tag{3}$$

$$T_{srv} \geq T_{neg} + T_{rsp} \tag{4}$$

where, T_{nneg} is the negotiated service time of the request which applied to enter the CDC and T_{srv} represents the total affordable service time.

Priorities of the requests in the queues are calculated and new VMs are created for requests with higher priority. At the end of serving a request, a VM becomes free and can serve requests if there are waiting requests in the queue of same type of requests. In this way, spending time for VM creation and deletion can be avoided and the system experiences a better performance. Requests get services according to their arrival time and time criticality. The queues are implemented using priority queue where the priority metric for each request is calculated using the following equation.

$$P_{fact}^{ij} = T_{ariv}^{ij} + T_{neg}^{ij}, \tag{5}$$

where, P_{fact}^{ij} is the priority factor of request i in queue j, T_{ariv}^{ij} represents the time of arrival of request i in the queue j and T_{neg}^{ij} is the negotiated service time for user request i in queue j. The priority of a queue is the summation of all the priorities of the requests it contains and the highest priority queue is denoted by j_{max}. The user request having highest priority in queue j_{max} is denoted by i_{max}. *Resource Allocator* provides new virtual machine for the user request i in queue j which has the maximum P_{fact}^{ij}.

Statement 1 of Algorithm 1 runs n times. Statement 12 and statements 16 $\sim$ 20 have a time complexity of $i \times j$. The rest of the statements of Algorithm 1 have unit time complexity. Therefore, the overall computation complexity of Algorithm 1 can be given by $MAX(n, i \times j)$.

The resource allocator of our proposed VSA system dynamically scales the number of VMs in the pool to execute user requests meeting SLA requirements and resource availability in the cluster. It can also migrate requests on-demand to neighboring clusters. Thus it is self-healing, distributed and self-adaptive in maintaining VMs of a VSA system.

4.3 Probabilistic analysis for admission control

As mentioned earlier, the system maintains a separate queue for each class of requests. Maximum queue size can be given by

$$Q_{max} = VMcreationtime \times \lambda_a, \tag{6}$$

where, λ_a is the average request arrival rate, the time-step $T = 1/\lambda_a$. Let, a be the arrival probability of a request in T and c be the VM allocation probability in a time-step. Thus, the system can be modeled as an $M^m/M/1/B$ queueing system with $b = 1 - a$, the probability that no requests arrives and $d = 1 - c$, the probability that no VM is allocated for the request. Assuming n_{max} is the maximum number of requests arrived in a time step, the binomial probability of arriving u requests can be obtained by (7).

$$a_u = \binom{n_{max}}{u} a^u b^{n_{max}-u} \tag{7}$$

The system must ensure that no request is allowed whenever the buffer is full, i.e., a stable system must abide by the following condition,

$$\sum_{u=0}^{n_{max}} u \times a_u = a \times n_{max} < c. \tag{8}$$

The state transition matrix for the system is given by

$$M = \begin{bmatrix} p & q_0 & 0 & \dots & 0 & 0 \\ q_2 & q_1 & q_0 & \dots & 0 & 0 \\ q_3 & q_2 & q_1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q_B & q_{B-1} & q_{B-2} & \cdots & q_1 & q_0 \\ r_B & r_{B-1} & r_{B-2} & \dots & r_1 & r_0 \end{bmatrix}$$

where, each entry M_{uv} implies the transition from state v to state u and

$$p = a_1 c + a_0 \tag{9}$$

$$q_u = a_u c + a_{u-1} d \tag{10}$$

$$r_u = a_u d + \sum_{k=u+1}^{m} a_k \tag{11}$$

Here, state represents the number of requests present in the queue. The throughput of our proposed queue for the state s_0 can be represented as

$$Th_0 = (1 - a_0) \times c \tag{12}$$

For all the other states, the throughput can be calculated as

$$Th_v = c \qquad 1 \leq v \leq B \tag{13}$$

The average throughput is the total number of VMs allocated to the requests in the queue and is estimated as

$$Th = \sum_{v=0}^{Q} Th_v s_v = c(1 - a_0 s_0) = N_a(out) \tag{14}$$

The input traffic of the system can be represented as

$$N_a(in) = \sum_{u=0}^{m} u \times a_u = a \times n_{max} \tag{15}$$

The efficiency of the proposed system is calculated as

$$\eta = \frac{N_a(out)}{N_a(in)} = \frac{Th}{n_{max} \times a} = \frac{c(1 - a_0 s_0)}{n_{max} \times a} \tag{16}$$

Since no request should be dropped after approval, the value of η should be equal to 1. That is, $N_a(out)$ should be equal to $N_a(in)$. Therefore,

$$c(1 - a_0 s_0) = n_{max} \times a \tag{17}$$

i.e., for ensuring QoS, *Admission Controller* should allow requests in such a way that values of n_{max} and a must maintain above equation of equilibrium.

5 Energy-efficient VM scheduling

In the case, usage of cluster resources falls below a predefined threshold, all of the requests in process and in queues of that cluster is migrated to an adjacent cluster. If a cluster is able to migrate all of its requests, we call it 'Power Saver (PS)' and one that agrees to serve those requests is called 'Neighbor Server (NS)'. For migrating requests from a PS, only a single PS is selected in the VSA system. This is because, dependent and co-related requests will spend a great amount of time in data transmission. Therefore, the system will experience improved performance in terms of response time since communication latency among the VMs can be avoided to a large extent. At first, the PS communicates with its neighbors and aks for occupancy ratio, μ. The equation for calculating μ is expressed as follows.

$$\mu = \frac{\phi_{ru}}{\phi_{ar}}, \tag{18}$$

where, ϕ_{ru} represents the total resource usage in a cluster and ϕ_{ar} represents the total resources available in that cluster.

The PS cluster now evaluates the μ value from all the neighboring clusters and selects one with minimum μ value. After migrating the running VMs to a NS cluster, the PS cluster can switch to *sleep* mode. Therefore, the cost of computation becomes less since a large amount of energy for operating the servers in a cluster can be saved. Whenever a PS cluster gets a new request while operating in *sleep* mode, all new requests of PS cluster are forwarded to the NS. In the case, execution load of NS crosses the maximum capacity, i.e., μ value crosses an upper threshold, it awakens the PS and migrates all VMs and requests. An NS cluster has to be active all the time even if its resource usage indicator μ value goes below the lower threshold value.

What follows, we present intra- and inter-cluster VM scheduling algorithms of the VSA system where computing resources are allocated to the requests in a distributed and energy-efficient way.

5.1 Scheduling algorithms

5.1.1 Intra-cluster resource management algorithm

We consider that, the physical servers will have different computation capability, i.e., the servers are heterogeneous in computation power and storage capacity. In case of VM migration, the workload ratio, ω, for each server is computed as follows,

$$\omega = \frac{\sigma_{tu}}{\sigma_{ta}} \times 100\%, \tag{19}$$

where, σ_{tu} denotes the total resource usage in a server and σ_{ta} denotes the total computing resources present in the server.

Resource Allocator will run an algorithm periodically to reduce the number of working servers within a cluster. *Resource Allocator* provides free VMs to the requests waiting in the input queues. At times, when the number of requests increases, workload ration (ω) increases proportionally. As a result, nearly all the servers operate in *active* mode. However, higher resource demand will not persist for every time period. If the number of working

servers is not reduced at lower workload period, huge amount of energy will get wasted. *Resource Allocator* tries to keep the number of active servers as few as possible by migrating the VMs from all the servers to a few active servers.

The *Resource Allocator* first calculates the value of occupancy ration (ω) for all servers within a cluster. Then it selects servers which have ω values smaller than a lower level threshold limit ω_{low}. It then migrates all the VMs from the selected servers to those servers which have a high ω and which can accommodate the whole VM. The destination servers where the VMs have been migrated, must have ω value lower than an upper threshold ω_{high} after accommodating the guest VMs. *Resource Allocator* keeps all the servers who get released of their computational operation in *sleep* mode. The steps of intra-cluster VM scheduling is summarized in Algorithm 2.

Algorithm 2 Intra-Cluster Resource Management Algorithm

INPUT: σ_{tu} and σ_{ta} *for all server machines*
OUTPUT: *State scheduling of server machines*

1: **for** each k in a cluster **do**
2: Calculate ω_k using Eq. 19
3: **end for**
4: **for** each server k in the cluster **do**
5: **if** $\omega_k < \omega_{low}$ **then**
6: $l \leftarrow \underset{l \in all\ servers}{\text{argmax}} \{\omega_k + \omega_l \leq \omega_{high}\}$
7: All VMs are migrated from k to l
8: Change state of k to sleep
9: **end if**
10: **end for**

Statements $1 \sim 3$ in Algorithm 2 run k times. Statements $4 \sim 10$ have a time complexity of k^2. The rest of the statements have unit time complexity. Therefore, the computation complexity of Algorithm 2 can be expressed as $O(k^2)$, where k is the number of servers in a cluster.

5.1.2 Inter-Cluster Resource Management Algorithm

At first, the value of the occupancy ratio μ is calculated for a cluster. Then it shares the value with its neighbors. Then, each cluster creates a Neighborhood Occupancy Vector (V_O) from the collected data. When the occupancy ratio of a cluster have a value smaller than μ_{low}, it selects a cluster in its neighborhood with minimum μ value. All the VMs from this cluster are migrated to the selected cluster by maintaining the constraint that μ value of the selected cluster must not exceed a upper threshold μ_{high}. Then the whole cluster operates in *sleep* mode. The steps of inter-cluster VM scheduling is summarized in Algorithm 3.

Whenever two VMs of two different clusters communicate continuously more than a pre-defined time period T_{com}, the two VMs are kept in the same cluster if resources are available. In this case, cluster with less occupancy ratio is selected. If it has resources available for the two communicating VMs, the VM in the cluster of less occupancy ratio is migrated.

Algorithm 3 Inter-Cluster Resource Management Algorithm

INPUT: *Resource usage occupancy ratio of the cluster* μ :, V_O : *Neighborhood occupancy vector*, N_{nbr} : *Number of neighbors of a cluster*
OUTPUT: *State scheduling of servers in a cluster*

1: Find out the value of μ using Eq. 18
2: **if** $\mu < \mu_{low}$ **then**
3: $\nu \leftarrow \underset{u \in ||V_O||}{\text{argmin}}\{\mu_u\}$
4: **if** $(V_O[\nu] + \mu) < \mu_{high}$ **then**
5: Migrate all VMs to cluster ν
6: Set $V_O[\nu] = V_O[\nu] + \mu$
7: Change the state of cluster to *sleep*
8: **end if**
9: **end for**

Statement 3 of Algorithm 3 has time complexity of $||V_O||$. The rest of the statements have unit time complexity. The overall time complexity of Algorithm 3 is $O(||V_O||)$.

5.2 Measurement of energy consumption

The energy consumption of each cluster can be calculated considering the current execution load of a cluster. Total energy conservation inside a cluster is calculated as

$$E_c(t) = (1 - \frac{\sum_{i=1}^{S_a} \omega i}{S_a \times 100}) \times E_{idle} + (S_n - S_a) \times E_{sleep}, \tag{20}$$

where, $E_c(t)$ represents the total amount of energy conservation of a cluster at time t, S_n and S_a represents the total number of servers and number of *active* servers in that cluster respectively. The first term of the equation represents the saved energy from servers operating in *idle* state. The second term represents the saved energy from servers in *sleep*. The total amount of energy conservation can be obtained by integration (20) over the desired period.

$$E_c = \int E_c(t)dt. \tag{21}$$

The total amount of energy consumption by a whole CDC considering resource utilization can be expressed as

$$E_i = N \times E_{max} \times M_{CC}, \tag{22}$$

where, N represents the number of clusters in the CDC; E_{max} represents the maximum amount of energy that could be consumed by a cluster when all the servers are operating in *active* state; and, M_{CC} is the mean computation cost, which is the average of resource utilization in all the clusters. Since M_{CC} is a fraction and value of M_{CC} is less than or equal to 1, $E_{max} \times M_{CC}$ provides the average of the energy consumption in each cluster of CDC. The M_{CC} can be given by

$$M_{CC} = \frac{\sum_{k=1}^{N} \mu_k}{N}. \tag{23}$$

These formulated equations of calculating energy consumption have been used for evaluating the performance of the VSA system and for comparing the results with the state-of-the-art works.

6 Performance evaluation

The performances of the our VSA system is evaluated in a distributed cloud computing environment using CloudSim toolkit [7]. The results presented in this section have been found from the comparative study of system performance of Energy-aware Hierarchical Scheduling (EHS) [36], Virtual Machine Provisioning Based on Analytical Performance and QoS (AVM) [6] and our proposed VM scheduling algorithms (VSA).

6.1 Simulation environment setup

We have considered a CDC having 5 clusters. Each of the clusters contains 4 servers. The servers are provided with different processing capacities from a list of processors including core-2-duo, core-i3, core-i5, and core-i7. Each server has 4GB, 8GB or 16GB RAM and multiple terabytes of secondary storage devices. Each core in the servers provide a performance equivalent to 3000 MIPS. The host servers within a cluster are connected by a communication link having 1 Gbps bandwidth and the clusters are connected by a link of 500 Mbps bandwidth.

ω_{low} and ω_{high} are considered to take values of 15 % and 85 % respectively. The values of μ_{low} and μ_{high} in each cluster are considered to be 20 % and 90 % respectively. For migration of communicating VMs, the value of T_{com}, described in Section 5.1.2 is considered to be two minutes. The number of request which arrive at the CDC is considered to be 30-120 per minute. Each request is considered to take time in the range 5-50 minutes to get served. For calculating energy consumption of the AVM system, we just observed the system energy usage after provisioning VMs to the requests solely based on analytical performance and QoS as illustrated in [6]. System energy is calculated by considering how many servers are working, how many VMs are communicating and resource utilization in each server. For measuring QoS of the EHS system, we simply implement the energy aware VM scheduling algorithm described in [36].

6.2 Performance metrics

- *Percentage of requests served:* The numbers of requests that can get service to the total number of incoming requests in a time step multiplied by hundred gives percentage of requests served in a time step. Since, higher number of requests served increases user satisfaction, it has been considered to evaluate the performance of QoS provisioning of the VSA system.
- *Mean VM creation time:* For evaluating our proposed mechanism for QoS provisioning, we have monitored the average time taken for creating a VM. Mean VM creation time increases as the number of VM increases.
- *Percentage of servers in sleep mode:* Severs in *sleep* mode consumes less energy. For this reason, we have calibrated the number of servers which can be kept in *sleep* mode for different arrival rate and for different resource requirement ratio.
- *Energy Consumption:* Energy consumption expresses the amount of energy that will be required to serve requests and energy consumed by VMs to communicate with each

other. Energy consumption is expressed as the ratio of energy required to serve user requests to the total amount of energy consumed to keep all the servers *active*.

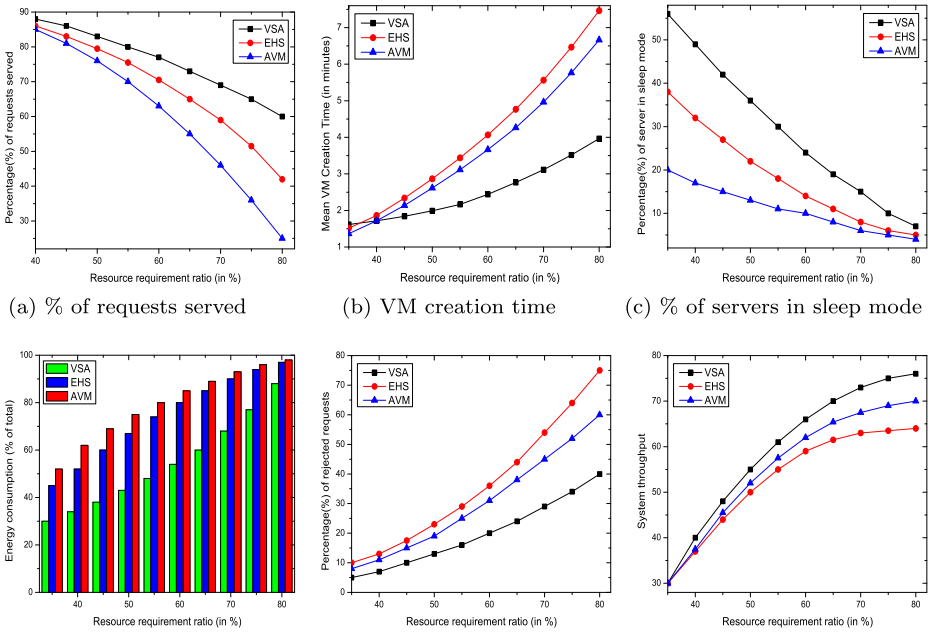
– *Percentage of rejected requests:* Increase in percentage of rejected requests will deteriorate the reputation of the CP. Percentage of rejected requests is calculated by taking the ratio of total number of rejected requests and total number of arriving requests and multiplying the ratio by hundred.
– *System Throughput:* System throughput is measured from the number of requests which get service in a unit time step. System throughput is also used to measure the effectiveness of a system, i.e., how effectively the system can server user requests within minimum time barrier.
– *Percentage of requests migrated:* Migration of requests to another cluster decreases system performance. As the number of migrated request increases, user satisfaction, user preference and reliability of the system decreases.

6.3 Simulation results

6.3.1 Impacts of varying resource requirement ratio

Figure 5 depicts system performance with respect to varying resource requirement ratio.

Figure 5a, shows the relation between percentage of requests served by AVM, EHS and VSA systems and varying resource requirement ratio. The figure reveals the fact that, as the request arrival rate increases, resource requirement ratio also increases proportionally. Therefore, the percentage of request served also decreases. From the figure we can see that,



(a) % of requests served (b) VM creation time (c) % of servers in sleep mode

(d) % of energy consumption (e) % of requests rejected (f) System throughput

Fig. 5 Impacts of resource requirement ratio on performances of the studied systems

as the resource requirement ratio increases, VSA system outperforms the existing AVM and EHS.

In Fig. 5b, a comparative study of mean VM creation time for varying resource requirement ratio is shown among the systems under study. The figure explains that, as the resource requirement ratio increases, necessity for creating new VMs for new requests also increases. As our proposed VSA recycles the existing VMs, time for creating new VM can be omitted and this time can be deployed to serve other user requests.

The comparative analysis among AVM, EHS and VSA systems on the percentage of servers that can be kept in *sleep* mode for varying resource requirement ratio is shown in Fig. 5c. The figure reveals the fact that, as the workload increases, the resource usage also increases linearly. Therefore, the number of servers, that can be kept sleeping also decreases. As VSA system accomplishes the intra- and inter- cluster scheduling algorithms, it can balance load among the working servers which enables it keep more servers sleeping compared to AVM and EHS.

Figure 5d shows the energy consumption for increasing workloads. Since inter- and intra-cluster VM scheduling algorithm has been used to minimize the number of working VMs in the VSA system, energy consumption gets reduced significantly. Again, VM migration has not only been considered at the time of request placement at physical servers in the VSA system but also at time of serving request while their demand changes. Thus, our algorithm of the minimization of energy consumption copes efficiently with the change of user demands from peak hour to off-peak hour. This increases the adaptability of the VSA system with the cloud DC system computation loads as well as the user demands. Our in-depth look into the contents of simulation trace file shows that, VSA system can keep considerably large number of servers in sleep mode and hence can save more energy compared to EHS and AVM approaches.

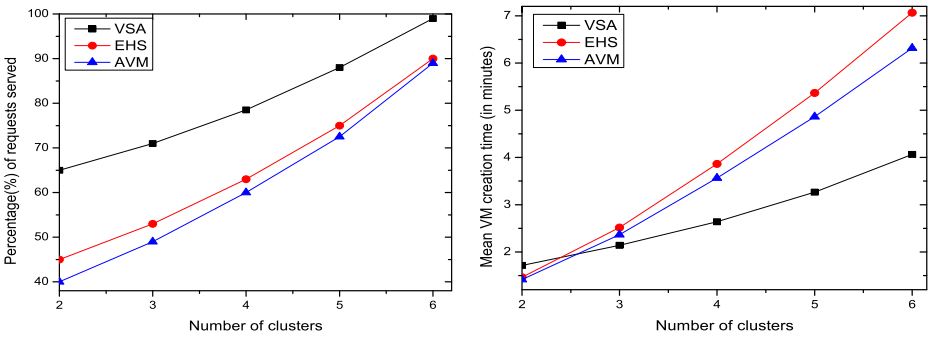
In the Fig. 5e, comparative study on the percentage of rejected requests between AVM, EHS and VSA systems for varying resource requirement ratio is shown. From the figure we can see that, as the workload increases, so is the resource requirement of the requests. Therefore, the percentage of request rejection is also increased. Since VM recycling is used in VSA system, time can be saved which can be deployed to serve more user requests. Hence, the rate of request rejection is much less compared to the existing AVM and EHS systems for VM allocation and scheduling.

Throughput for varying resource requirement ratio implementing different VM provisioning and scheduling approaches is depicted in Fig. 5f. The figure reveals the fact that, throughput obtained by applying our proposed VSA is higher compared to the existing EHS and AVM. This is because, VSA saves time by VM recycling which is used to serve more requests which increases the system throughput.

6.3.2 System performances for varying number of clusters

Figure 6 shows the system performance for varying resource requirement ratio. In the Fig. 6a a comparative study on the percentage of requests that can be served in the CDC through varying number of clusters is shown. The figure depicts the fact that, the percentage of served requests increases with the increase of number of cluster. However, as VSA system reuses created VMs through dynamically managing VM pool, it can serve more requests compared to EHS and AVM system for the same number of active clusters.

Figure 6b shows the relation between mean time required for VM creation and number of active clusters in a data center. The figure reveals the fact that, as the number of



(a) % of requests served (b) VM creation time

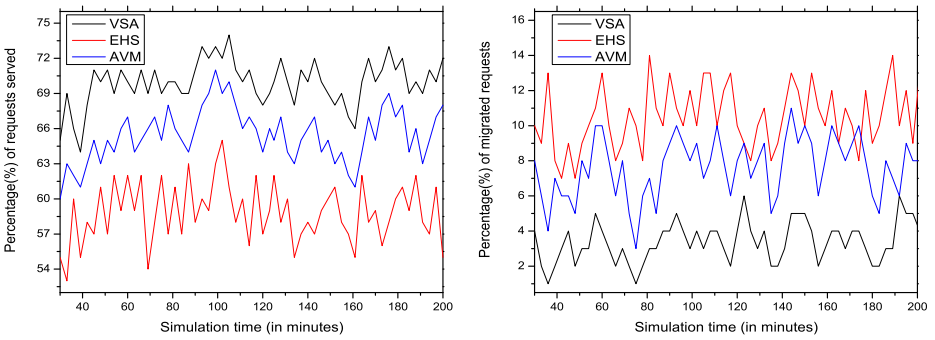
Fig. 6 Impacts of varying cluster numbers on system performance

cluster increases, the capacity for creating new VMs for serving increasing number of user requests also increases. As a result, mean time for VM creation also increases for serving the additional user requests. Since our proposed VSA system recycles the created VMs, it can minimize the time for creating VMs and hence it can utilize the saved time to serve additional requests.

6.3.3 System performances during simulation period

Figure 7 shows the system performance for varying simulation period. In the Fig. 7a, the comparative study on percentage of request served with time among EHS, AVM and VSA systems is given. The figure reveals the fact that, more number of user requests get served in VSA system compared to AVM and EHS. This is due to the fact that, VM creation and deletion time in the VSA system can be eliminated and hence profit and user satisfaction get maximized.

In the Fig. 7b, the percentage of requests migrated to another cluster with time is presented. The performance comparison is again made among AVM, EHS and VSA systems. From the figure we can see that, less number of requests need to be migrated by applying



(a) % of served requests (b) % of migrated requests

Fig. 7 Impacts of simulation time on served and migrated requests

VSA. During the peak hours, VSA serves more user requests which decreases the migration rate. As a result, user satisfaction along with resource utilization increases. Therefore, profit of CPs also rises.

6.4 Discussion

A simple modified version of our proposed VSA system can also operate in a self-adaptive mobile cloud computing system. In the mobile cloud, a number of mobile computing terminals create a temporary cluster and select a broker. The broker node runs the modified VSA system to decide dynamically whether to accept, reject or migrate a request.

This paper designs a QoS aware VM provisioning algorithm that adaptively schedules VMs among the available machines in an energy-efficient manner. We have used the response time, the average time required to serve a request, as the QoS parameter and methodologies have been developed to satisfy it.

User satisfaction increases in a CDC system if the SLA requirements are not violated. While allocating VMs, we consider system congestion and resource availability to receive new user requests. As a result, the VSA system can guarantee that the admitted requests are executed without violating SLA. Therefore, the QoS is increased by serving user requests faster than the time specified in the SLA.

7 Conclusion

This work explores the ways of reducing energy consumption costs while keeping user Service Level Agreements (SLAs) in a ubiquitous social multimedia cloud computing environment. A novel queuing model for customer requests and an energy-efficient scheduling mechanism for allocating VMs on physical servers are developed. The system components are designed to efficiently process the data from social media cloud applications that require heterogeneous services. Our performance evaluation, carried out in CloudSim, shows that the proposed VSA system achieves significant performance improvements in terms of user satisfaction, energy conservation, and system throughput in comparison to other related studies.

Acknowledgments This project was full financially supported by the King Saud University, through Vice Deanship of Research Chairs.

References

1. Adhikary T, Das AK, Razzaque MA, Sarkar AMJ (2013) Energy-efficient Scheduling Algorithms for Data Center Resources in Cloud Computing, IEEE HPCC 2013, Zhangjiajie, China
2. Adhikary T, Das AK, Razzaque MA, Almogren A, Alrubaian M, Hassan MM (2015) Quality of Service Aware Reliable Task Scheduling in Vehicular Cloud Computing, Journal of Mobile Network and Applications (MONET), Springer. doi:10.1007/s11036-015-0657-5
3. Baliga J, Ayre RWA, Hinton K, Tucker RS (2011) Green cloud computing: Balancing energy in processing, storage, and transport. Proc IEEE 99(1):149–167
4. Beloglazov A, Buyya R (2010) Energy efficient resource management in virtualized cloud data centers, 10th IEEE/ACM International Conference on Cluster Cloud and Grid Computing
5. Bi J, Zhu Z, Tian R, Wang Q (2010) Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center Proceedings of the 3rd International Conference on Cloud Computing (CLOUD'10)

6. Calheiros RN, Ranjan R, Buyya R (2011) Virtual machine provisioning based on analytical performance and QoS in cloud computing environments international conference on parallel processing (ICPP)
7. Calheiros R, Ranjan R, Beloglazov A, Rose C, Buyya R (2011) Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Experience 41(1):23–50
8. Chang HM, Chao HC, Chen JL, Lai CF (2012) An efficient service discovery system for dual-stack cloud file service. IEEE Syst J 6(4):584–592
9. Das AK, Adhikary T, Razzaque MA, Hong CS (2013) An Intelligent Approach for Virtual Machine and QoS Provisioning in Cloud Computing, International Conference on Information Networking ICOIN, Bangkok, Thailand
10. Das AK, Adhikary T, Razzaque MA, Cho EJ, Hong CS (2014) A QoS and profit aware cloud confederation model for IaaS service providers, Proceedings of ACM IMCOM 2014, Siem Reap, Cambodia
11. Djemame K, Haji MH (2007) Grid application performance prediction: a case study in BROADEN the first international workshop on verification and evaluation of computer and communication systems (VECos)
12. Emeneker W, Apon A (2012) Characterising the performance of cache-aware placement of virtual machines on a multi-core architecture. Int J Ad Hoc Ubiquitous Comput, 2012 10(2):84–95
13. Fang Q, Sang J, Xu C, Hossain MS (2015) Relational user attribute inference in social media. IEEE Trans Multimedia 17(7):1031–1044
14. Gain B (2016) Cloud Computing & SaaS In 2010 Processor Magazine. http://www.processor.com/editorial/article.asp?article=articles/P3201/23p01/23p01.asp&guid=
15. Hassan MM (2014) Cost-effective resource provisioning for multimedia cloud-based e-health systems Springer multimedia tools and applications
16. Hassan MM, Song B, Hossain MS, Alamri A (2014) Efficient virtual machine resource management for media cloud computing. KSII Trans Internet Inf Syst 8(4):1567–1586
17. Hefeeda M, ElGamal T, Calagari K, Abdelsadek A (2015) Cloud-based multimedia content protection system. IEEE Trans Multimedia 17(3):420–433
18. Hossain MS, Muhammad G (2016) Cloud-assisted industrial internet of things (IoT)?enabled framework for health monitoring. Comput Netw 101:192–202
19. Hossain MS, Muhammad G, Alhamid MF, Song B, Almutib K (2016) Audio-visual emotion recognition using big data towards 5G. Mobile Networks and Applications. doi:10.1007/s11036-016-0685-9
20. Kaplan J, Forrest W, Kindler N (2008) Revolutionizing data center energy efficiency, McKinsey & Company, Tech. Report
21. Knauth T, Fetzer C (2012) Energy-aware scheduling for infrastructure clouds IEEE 4th international conference on cloud computing technology and science (CloudCom)
22. Lai CF, Wang H, Chao HC, Nan G (2013) A network and device aware QoS approach for cloud-based mobile streaming. IEEE Trans Multimedia 15(4):747–757
23. Lee YC, Wang C, Zomaya AY, hou BB (2010) Profit-driven service request scheduling in clouds, Proceedings of the 10th IEEE/ACM International Conference on Cluster Cloud and Grid Computing (CCGrid'10)
24. Liu L, Dasilva DA, Antonopoulos N, Ding Z, Zhan Y (2013) Achieving Green IT using VDI in cyber physical society. J Internet Technol 14(3):413–424
25. Lu P, Sun Q, Wu K, Zhu Z (2015) Distributed online hybrid cloud management for Profit-Driven multimedia cloud computing. IEEE Trans Multimedia 17(8):1297–1308
26. Mahmud MR, Afrin M, Razzaque MA, Hassan MM, Alelaiwi A, Alrubaian M (2016) Maximizing quality-of-experience through context aware mobile application scheduling in cloudlet infrastructure, Software: Practice and experience, Wiley InterScience spe2392
27. Moreno IS, Xu J (2011) Customer-aware resource overallocation to improve energy efficiency in real-time Cloud Computing data centers IEEE International Conference on Service-Oriented Computing and Applications (SOCA)
28. Nathuji R, Schwan K (2007) Virtualpower: Coordinated power management in virtualized enterprise systems. ACM SIGOPS Oper Syst Rev 41(6):265–278
29. Oberheide J (2008) Exploiting live virtual machine migration, University of Michigan, Black Hat DC
30. Rodero-Merino L, Vaquero LM, Gil V, Galan F, Fontan J, Montero RS, Llorente IM (2010) From infrastructure delivery to service management in clouds. Futur Gener Comput Syst 26(8):1226–1240
31. Shy O (2011) Overbooking, How to Price. Cambridge University Press, Cambridge
32. Velte AT Chapter One: Cloud Computing Basics, Cloud Computing: A Practical Approach ed: McGraw-Hill, 3–22

33. VMware (2009) VMware Distributed Power Management Concepts Use, VMware Inc, Palo alto, CA, USA, Tech. Rep IN-073-PRD-01-01
34. Wang L, Chen D, Zhao J, Tao J (2012) Resource management of distributed virtual machines. Int J Ad Hoc Ubiquitous Comput 10(2):96–111
35. Wen Y, Zhu X, Rodrigues JJPC, Chen CW (2014) Cloud mobile media: Reflections and outlook. IEEE Trans Multimedia 16(4):885–902
36. Wen G, Hong J, Xu C, Balaji P, Feng S, Jiang P (2011) Energy-aware hierarchical scheduling of applications in large scale data centers International Conference on Cloud and Service Computing (CSC)
37. Yang X, Zhang T, Xu C, Hossain MS (2015) Automatic visual concept learning for social event understanding. IEEE Trans Multimedia 17(3):346–358
38. Zhang P, Yan Z (2011) A QoS-Aware System for Mobile Cloud Computing Proceedings of IEEE (CCIS)
39. Zeng H, Ellis CS, Lebeck AR, Vahdat A ECOSys-tem: managing energy as a first class operating system resource. ACM SIGPLAN Not 37(10):132

Tamal Adhikary received his BS and MS from the Department of Computer Science and Engineering, University of Dhaka, Bangladesh in 2013 and 2015, respectively. He is a research assistant of the Green Networking Research Group of the department. His research interests include Cloud Computing, Mobile Cloud Computing, Green Cellular Network, Vehicular Cloud, etc.

Amit Kumar Das received his BS and MS from the Department of Computer Science and Engineering, University of Dhaka, Bangladesh in 2013 and 2015, respectively. He is a research assistant of the Green Networking Research Group of the department. His research interests include Big Data Cloud, Cloud Computing, Mobile Cloud Computing, Green Cellular Network, etc.

Md. Abdur Razzaque received his B.S. and M.S. degrees from the University of Dhaka, Bangladesh and he obtained his PhD degree in Wireless Networking from Kyung Hee University, South Korea in August, 2009. He was a research professor, College of Electronics and Information, Kyung Hee University, South Korea during 2010-2011. He is now working as a Professor in the Department of Computer Science and Engineering, University of Dhaka, Bangladesh. He is the group leader of Green Networking Research Group (http://gnrbd.net) of the same department. His research interest is in the area of modeling, analysis and optimization of wireless networking protocols and architectures, Wireless Sensor Networks, Body Sensor Networks, Cooperative Communications, Sensor Data Clouds, Cognitive Radio Networks, etc. He has published a good number of research papers in IEEE/ACM/Springer conferences, journals, and books. He is TPC members of IEEE HPCC, ICOIN, ADM, ICUFN, NSyS, etc. He is a senior member of IEEE, IEEE Communications Society, IEEE Computer Society, Internet Society (ISOC), Pacific Telecommunications Council (PTC) and KIPS.

Majed Alrubaian is currently a PhD student in the Department of Information Systems in the College of Computer and Information Sciences (CCIS), King Saud University (KSU), Riyadh, Kingdom of Saudi Arabia. His research interests include mining social data, cloud computing and sensor network. He is a student member of IEEE and ACM.

Mohammad Mehedi Hassan is an Assistant Professor of Information Systems Department in the College of Computer and Information Sciences, King Saud University, Riyadh, Kingdom of Saudi Arabia. He received his Ph.D. degree in Computer Engineering from Kyung Hee University, South Korea in February 2011. He was a Research Professor at Computer Engineering department, Kyung Hee University, South Korea from March, 2011 to October, 2011. He has authored and co-authored more than 50 publications including refereed IEEE/ACM/Springer journals, conference papers, books, and book chapters. He has served as, chair, and Technical Program Committee member in numerous international conferences/workshops like IEEE HPCC, IEEE ICME, ACM Multimedia, ICA3PP, IEEE ICC, TPMC, IDCS, etc. His research interests include Cloud collaboration, multimedia Cloud, sensor-Cloud, mobile Cloud, Thin-Client, Grid computing, IPTV, virtual network, sensor network, and publish/subscribe system.

Atif Alamri is an Associate Professor of Information Systems Department in the College of Computer and Information Sciences, King Saud University, Riyadh, Kingdom of Saudi Arabia. He received his Ph.D. degree from School of Information Technology and Engineering, University of Ottawa, Canada in 2010. He completed his BS and MS degrees from College of Computer and Information Science in King Saud University, Kingdom of Saudi Arabia. He has authored and co-authored more than 40 publications including refereed IEEE/ACM/Springer journals, conference papers, books, and book chapters. He has served as, chair, and Technical Program Committee member in numerous international conferences/workshops. His research interests include Cloud computing, Sensor networks, multimedia Cloud, sensor-Cloud, mobile Cloud, etc.