

# A two-stage approach for task and resource management in multimedia cloud environment

Biao Song · Mohammad Mehedi Hassan · Atif Alamri ·  
Abdulhameed Alelaiwi · Yuan Tian · Mukaddim Pathan ·  
Ahmad Almogren

Received: 12 January 2014 / Accepted: 20 May 2014 / Published online: 4 June 2014  
© Springer-Verlag Wien 2014

**Abstract** In recent years, multimedia cloud computing is becoming a promising technology that can effectively process multimedia services and provide quality of service (QoS) provisioning for multimedia applications from anywhere, at any time and on any device at lower costs. However, there are two major challenges exist in this emerging computing paradigm: one is task management, which maps multimedia tasks to virtual machines, and the other is resource management, which maps virtual machines (VMs) to physical servers. In this study, we aim at providing an efficient solution that jointly addresses these challenges. In particular, a queuing based approach for task management and a heuristic algorithm for resource management are proposed. By adopting allocation deadline in each VM request, both task manager and VM allocator receive better chances to optimize the cost while satisfying the constraints on

---

B. Song · M. M. Hassan (✉) · A. Alamri · A. Alelaiwi · Y. Tian · A. Almogren  
College of Computer and Information Sciences and Research Chair of Pervasive and Mobile Computing,  
King Saud University, Riyadh 11543, Kingdom of Saudi Arabia  
e-mail: mmhassan@ksu.edu.sa

B. Song  
e-mail: bsong@ksu.edu.sa

A. Alamri  
e-mail: atif@ksu.edu.sa

A. Alelaiwi  
e-mail: aalelaiwi@ksu.edu.sa

Y. Tian  
e-mail: ytian@ksu.edu.sa

A. Almogren  
e-mail: aahmogren@ksu.edu.sa

M. Pathan  
Telstra Corporation Limited, 10/35 Collins St, Melbourne, VIC 3000, Australia  
e-mail: mukaddim.pathan@team.telstra.com

the quality of multimedia service. Various simulations were conducted to validate the efficiency of the proposed task and resource management approaches. The results showed that the proposed solutions provided better performance as compared to the existing state-of-the-art approaches.

**Keywords** Multimedia · Cloud computing · Task management · Resource management · Queuing system · Heuristic algorithm

**Mathematics Subject Classification** 60K25 Queueing theory · 60K30 Applications (congestion, allocation, storage, traffic, etc.)

## 1 Introduction

With the rapid growth of Web 2.0, much of the multimedia services has been delivered over Internet. The online multimedia systems consist of a rich number of operations including generating, editing, processing, searching and saving media contents. Strong and heterogeneous demands on computation, storage and communication resources have been involved in supporting such systems [1]. During the past decade, cloud computing has been recognized as a promising infrastructure to provide desired resources and quality of service (QoS) provisioning for multimedia services [1–4]. Various cloud based multimedia applications can be found in recent years, such as cloud based photo and video sharing, online photo and video editing, multimedia social applications and so on [5–8]. Cloud computing is designed to offer QoS guaranteed, on-demand and cost effective hardware and software solutions with the advances of virtual machine (VM) technology.

Two of the major research issues in cloud based multimedia context are the task management problem [7, 9–12] and the resource management problem [13–22]. On the one hand, task management is achieved by allocating existing resources, requesting new resources or releasing redundant resources based on the fluctuation of workload. Two important concerns in the task management studies are QoS and cost. On the other hand, resource management is solved by adopting VM allocation approach. i.e., whenever the VM requests are made, the VM allocator deals with the mapping of VM requests to physical resources where the VM instances reside. The optimization objective of cloud VM allocation usually contains two parts: long-term cost and waiting time.

Although there exist several studies regarding cloud based multimedia systems, the two research issues we mentioned above have not been jointly considered. For task management, queuing system is widely used for task management since the parameters of queue has direct connection with the QoS and the cost of multimedia services [9, 23]. For resource management, heuristic based VM allocation approaches are frequently adopted because of the problem complexity [14, 15]. It is very often that the efficiency of those solutions are measured individually. However, the task management approach should be compatible with the resource management approach, especially when both multimedia service provider and cloud operator want to adopt their own management policy. Seeing the problem from this perspective inspires us to a new solution which jointly supports both task management and resource management in a flexible and efficient way.

In this paper, we present an innovative framework that addresses the task and resource management challenges in multimedia cloud computing. To manage multimedia tasks, multiple queues are maintained by evaluating the performance indicators such as queue length, task blocking probability and mean service waiting time. The task manager of each queue creates VM requests or stops running VMs based on the queuing dynamics. The requested VM have multi-dimensional resource specification including CPU processing capability, memory space, storage space and network bandwidth. Besides, we propose to add allocation deadline for each VM request since immediate allocation stands in the way of cost reduction of both multimedia service provider and cloud operator. For multimedia service provider, the unpredicted fluctuation of task arrival rate may result in extra cost caused by fetching and stopping VMs frequently and unnecessarily. After receiving the VM requests, the focus of cloud operator is to allocate the VMs to physical cloud servers. For cloud operator, the chance of finding better allocation solutions is relatively low if all the VM requests has to be allocated immediately. The main contributions of this paper are listed as follows:

- We jointly design a two-stage task and resource management approach for cloud based multimedia services. Our solution allows both multimedia service provider and cloud operator to apply their own management policy without much interference. Meanwhile, the QoS of services are guaranteed through the proposed two-stage process. Thus, it is suitable to be adopted in large-scale and heterogeneous cloud based multimedia system.
- As it is not possible to provide exact modeling for heterogeneous multimedia services, we use an queuing model to approximate the system states. Based on the queuing model, an online and efficient task management algorithm using Markov analysis and prediction is proposed in this paper. The concept of allocation deadline differs our proposal from existing methods and benefits both multimedia service provider and cloud operator. In task management stage, the VM requests having allocation deadline can achieve cost reduction by preventing unnecessary fetching of VMs from happening while satisfying the QoS constraint.  
In order to guarantee the QoS of multimedia services, the following methods are used. First of all, the resource requirement in the QoS can be found by profiling multimedia applications. It will decide the configuration of required VMs in our proposed queuing model. Secondly, the deadline of application finish time will be fulfilled by controlling service time or average queue length in queuing model. Meanwhile, the blocking probability in queuing model is also a part of the QoS requirement which can also be satisfied by managing queuing dynamic.
- We propose an online and efficient heuristic based resource management approach for supporting cloud based multimedia service. The main difference between the algorithm in our work and conventional VM allocation algorithms is that our algorithm leverages the benefit of allocation deadline without taking any risk on QoS violation. Thus, it is possible to select and allocate task in an active and dynamic way through dynamic controller, while conventional task allocation algorithms perform in a passive or static way.
- Last but not the least, we prove that both task manager and resource allocator have the capability of stabilizing the QoS parameters for multimedia services by

sacrificing the cost. This feature enables flexible management policy design when dynamic pricing has been adopted by cloud operator.

Our approach also provides elasticity to resource provision of cloud multimedia applications. On one hand, the queuing model can automatically evaluates and requests resources to support dynamic scaling for distributed system. On the other hand, the dynamic scaling of a certain VM can be achieved with current VM technologies. The VM migration technology will be applied if the above scaling causes physical server overload. Since cloud provider always maintains service level agreements (SLAs) with application providers, migration decision has to be made based on SLAs. Although the SLAs are out of the scope of this paper, our approach can treat the VMs to be migrated as new VM requests and provide allocation for them.

Analytical results of task management are validated through discrete-event simulation. More experiments are conducted to validate the efficiency of our proposed resource management algorithm as well. The experimental input is produced based on cloud implementation of practical multimedia services. We compare our proposed algorithm with other existing algorithms, and present results as well as insights from the evaluation results in this paper.

The rest of the paper is organized as follows: Related work is reviewed in Sect. 2. Section 3 describes the general framework of task and resource management in cloud based multimedia system, and proposes task management solution. Section 4 presents the proposed online VM allocation algorithm, and discusses the stability of QoS. The experiment settings and performance comparisons can be found in Sect. 5. Finally, Sect. 6 concludes the paper.

## 2 Related work

In recent years, the concept of cloud based multimedia system is emerging and growing rapidly. The cloud computing has been investigated to provide multimedia services in various ways. Firstly, cloud can be used to store multimedia contents. In [6], the authors proposed a framework for scalable cloud video recorder system in surveillance environment. Hadoop distributed file system was applied to store video data. Other distributed systems designed in [24–26] are also very suitable for supporting multimedia cloud due to their significant contributions on handling the heterogeneity and uncertainty of multimedia tasks. Secondly, cloud is capable of providing scalable and distributed processing power to achieve efficient video analytic. In [5], a novel system was invented to bringing together automatic license plate recognition engines and cloud computing technology in order to realize massive data analysis and enable the detection and tracking of a target vehicle in a city with a given license plate number. Real-time face recognition approach was implemented using a mobile-cloudlet-cloud acceleration architecture in [8]. In another study [27], multiclass object recognition using smart phone and cloud computing for augmented reality was invented. The variety of multimedia tasks has brought two challenging issues to cloud provider, i.e., task management and resource management. To address these challenges, various of modeling and algorithms have been proposed.

## 2.1 Task management for cloud based multimedia system

Currently there are also some research efforts going on [7,9–12,28,29] for allocating VM resources in a multimedia cloud environment, which can be viewed as task management. Wen et al. [12] presented an effective load-balancing algorithm for a cloud-based multimedia system, which can allocate and schedule VM resources for different user requests with minimum costs. In [30], the authors considered the camera workload as a function of the number of targets, and use that to dynamically assign video feeds to the hosts.

Queuing models has been adopted to provide rigorous analytical approach in cloud task management domain. Nan et al. [9] proposed a cost effective resource allocation optimization approach for multimedia cloud that was based on a queuing model. They considered the data center infrastructure as a node-weighted tree-like graph, and then used the queuing model to capture the relationship between the service response time and the allocated resources. They also studied the resource allocation problem in a single-class service case and a multiple-class service case, respectively. The same authors also present similar approach in [10,11].

In [31], the authors explicitly used a  $M/M/m/m + r$  queuing model to manage tasks in a cloud center. Both interarrival and service times were assumed to be exponentially distributed. Since interarrival time and/or service time may not always follow exponential distribution, most theoretical analyses have relied on research in performance evaluation of  $M/G/m$  queuing system [32] or  $M/G/m/m + r$  queuing system [23,33]. Those works only provided approximate solutions for steady-state queue length distribution, average queuing delay and blocking probability. As we introduce the concept of allocation deadline, our modeling differs from exiting methods by identifying and predicting a specific moment when QoS violation will occur in queuing system. Consequently, we need to accomplish corresponding analyses which cannot be found in exiting works.

## 2.2 Resource management for cloud based multimedia system

On the other hand, many existing efforts [13–22] studied various VM allocation techniques for cloud resource management. Aisopos et al. [13] proposed a VM resource allocation model for SaaS cloud providers using fractional knapsack problem that maximizes the service provider's revenue and the resource utilization under a heavy load. The model focused on maintaining the maximum resource utilization at the cost of risking potential SLA violations over the pending jobs that will yield the smaller profit for the SaaS cloud provider. Van et al. [19] presented an autonomic virtual resource management mechanism in cloud for service hosting platforms. Game-theory based VM resource management for clouds was also studied in [17,22]. To reduce the cost for cloud provider, researchers focused on energy saving while allocating VMs. Beloglazov et al. [14] investigated energy-aware VM resource allocation heuristics that provided data center resources to client applications in a way that improved the energy efficiency of the data center without violating the negotiated SLAs. In [34], scheduling heuristics were developed to present application experience for reducing power con-

sumption of parallel tasks in a cluster with the Dynamic Voltage Frequency Scaling (DVFS) technique.

Although the works presented above have made remarkable contributions, they did not jointly consider both task and resource management. As a result, it is difficult to achieve an optimization for the entire system. In this paper, we address this issue by developing a two stage-solution to solve both problems.

### 3 Cloud-based multimedia system and task management

Cloud-based multimedia system utilizes powerful Cloud resources and provides rich media services to the users over the Internet. Figure 1 depicts the framework of cloud based multimedia system.

#### 3.1 Cloud based multimedia system

Several multimedia services, such as gaming, stream mining, video playback, multimedia surveillance and etc., are provided by the system. Each service transmits its tasks to one or many functional modules. In each module, the arrived tasks are gather in the corresponding job queue and processed by one or many virtual machines using the principle of First Come First Serve (FCFS). Each module also takes the responsibility of submitting, canceling or changing virtual machine requests, and shutting down redundant virtual machines. Through formulating and analyzing the queuing model in each module, the number of running virtual machine are adjusted dynamically to guarantee the quality of service and to reduce the cost for holding or fetching virtual machines. These modules can be designed in either centralized or distributed way to support multimedia services or to improve their efficiency. For example, distributed file systems or large-scale video bases are deployed to store massive amount

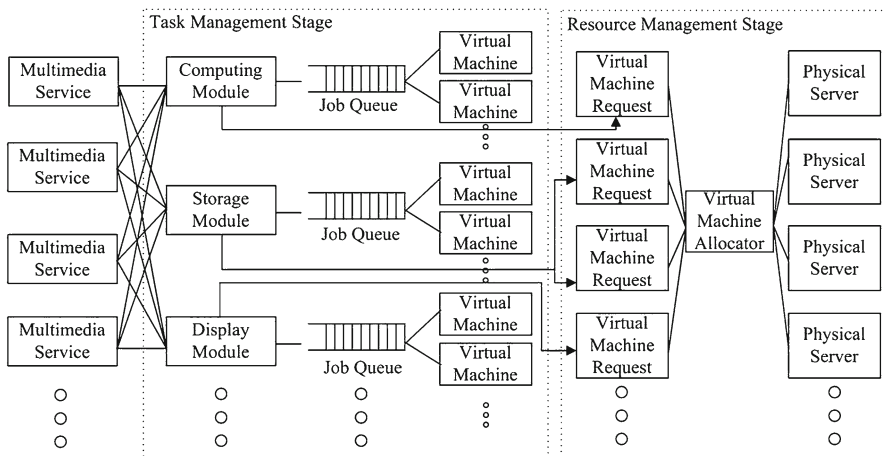


Fig. 1 Framework of cloud based multimedia system

of video clips for large scale surveillance tasks. The parameters used by each module for making their task management decision includes current queue length, recent task inter-arrival time in average and estimated service time in average.

To provision processing, storage, and networking resources, cloud provider offers Infrastructure as a Service (IaaS) to each module by implementing operating system level virtualization technology such as VMware and Xen hypervisors. To be more specific, each module can get CPU, memory, hard-disk and network bandwidth resources from cloud servers by submitting explicit virtual machine requests along with allocation deadline. The virtual machine allocator adopts a dynamic allocation approach to find an appropriate physical server according to each virtual machine request. In order to guarantee the QoS of multimedia services, the VM allocation has to at least meet the resource and deadline requirements. Besides, the allocator may also apply its own policy to achieve a global optimization on a desired tradeoff between long-term cost and average waiting time. In this paper, we formulate a time-slotted problem with time slots of equal length indexed by  $t = 0, 1, \dots$ . At the end of each time slot  $t$ , task management decision is made by each module first. After that, the VM allocator makes its resource management decision. Although different optimization methods and policies are used in two management stages, the QoS of multimedia services can be guaranteed through the whole process.

### 3.2 Analytical model for task management

In this study, each module is modeled as a  $M/G/m/m+r$  queuing system indicating that the inter-arrival time of tasks follows exponential distribution, while  $m$  virtual machines are deployed and the service times are independent and identically distributed random variables. We consider that the service times follow a general distribution. Any new task arrival will be discarded if the queue length is  $m+r$ . All tasks are served following the order of their arrival sequence (FCFS). This  $M/G/m/m+r$  queuing system can be considered as an embedded-Markov process [35] since the arrivals of tasks form a Markovian process [36]. Each arrival is a Markov point for us to observe the state of the system, which is the number of the tasks in the system whose value is a integer ranging from 0 to  $m+r$ . The assumption we are making in the queuing system for task management, including the distribution of interarrival time, the distribution of service time and the existence of steady-state, has been justified by existing literatures [23]. However, our assumption of considering allocation deadline makes significant difference in the results. Many real world multimedia applications require SLA on the finish time rather than how VMs are allocated over time. For example, in e-health media application scenario some tasks should follow their deadlines [37]. In our model, such tasks will be executed in a queuing system. The task manager will calculate the service time for such tasks based on current queuing dynamic. By analyzing the queuing dynamic, the task manager can predict overload situation in future. Thus, VM requests with allocation deadline will be generated to prevent overload situation from happening. Eventually, the SLA of multimedia tasks can be fulfilled.

The task inter-arrival time  $U$  is exponentially distributed with a rate of  $\lambda$ . The Cumulative Distribution Function (CDF) can be denoted as  $U(x) = \text{prob}[U < x]$ .

We denote its probability density function (pdf) as  $u(x) = \lambda e^{-\lambda x}$  and its Laplace Stieltjes Transform (LST) as  $U^*(s) = \int_0^\infty e^{-sx} u(x) dx = \frac{\lambda}{\lambda + s}$ . All task service times follow a general distribution  $V$  with a mean value of  $\bar{v} = \frac{1}{\mu}$ . The CDF and pdf of  $V$  are  $V(x) = \text{prob}[V < x]$  and  $v(x)$ , respectively. The LST of  $V$  is  $V^*(s) = \int_0^\infty e^{-sx} v(x) dx$ . We use  $V_+$  to denote residual task service time and  $V_-$  to denote elapsed task service time. According to [38], both  $V_+$  and  $V_-$  have the same LST, which is  $V_+^*(s) = V_-^*(s) = \frac{1 - V^*(s)}{s\bar{v}}$ .

We are able find one-step transition probabilities for above Markov model. Let  $q_n$  and  $q_{n+1}$  be the number of tasks found in the system before  $n$ th arrival and  $(n + 1)$ th arrival, respectively. The transition probability  $p_{ij}$  is defined as

$$p_{ij} = \text{prob}[q_{n+1} = j | q_n = i] \tag{1}$$

where  $q_{n+1} = j$  and  $q_n = i$  indicate that  $i + 1 - j$  tasks are accomplished during the time between the arrival of task  $n$  and the arrival of task  $n + 1$ .

In order to calculate one-step transition matrix  $P$ , we need to find each element  $p_{ij}$  by analyzing the probability of task departures in between two successive arrivals. Let us choose any server running on a virtual machine and focus on that server. For a running task to finish between two Markov points, the residual service time  $V_+$  should be less than the task inter-arrival time  $U$  with the probability of

$$\begin{aligned} P_x &= \text{prob}[V_+ < U] = \int_{x=0}^\infty \text{prob}[V_+ < U | V_+ = x] dV_+(x) \\ &= \int_{x=0}^\infty \left( \int_{y=x}^\infty \lambda e^{-\lambda y} dy \right) dV_+(x) = V_+^*(\lambda) \end{aligned} \tag{2}$$

If the entire processing of a task is finished during the time between the arrival of task  $n$  and the arrival of task  $n + 1$ , it holds the probability of

$$\begin{aligned} P_y &= \text{prob}[V < U] = \int_{x=0}^\infty \text{prob}[V < U | V = x] dV(x) \\ &= \int_{x=0}^\infty \left( \int_{y=x}^\infty \lambda e^{-\lambda y} dy \right) dV(x) = V^*(\lambda) \end{aligned} \tag{3}$$

The one-step transition matrix  $P$  can be calculated using the following methods:

- For  $i + 1 < j$ ,  $p_{ij} = 0$ .
- For  $i < m$  and  $j \leq m$ , the number of waiting tasks is 0. In one possible scenario, the newly arrived task and  $i - j$  running tasks has been accomplished. In the other possible scenario,  $i + 1 - j$  running tasks departs from the system. Thus, the probability is



$$p_{ij} = \binom{i}{i-j} P_x^{i-j} (1 - P_x)^j P_y + \binom{i}{i+1-j} P_x^{i+1-j} (1 - P_x)^{j-1} (1 - P_y)$$

for  $i < m, j \leq m$  (4)

- For  $i, j \geq m$ , the newly arrived task cannot be finished as all servers are busy. However, it is possible that more than one task depart from one server. Since possibility of finishing three or more tasks between successive arrivals is very low, we assume that no more than two tasks will depart from any server, the one-step transition probability can be denoted as

$$p_{ij} = \sum_{k_1=\max(\lceil(i+1-j)/2\rceil,0)}^{\min(i+1-j,m)} \left\{ \binom{m}{k_1} P_x^{k_1} (1 - P_x)^{m-k_1} \cdot \binom{k_1}{i+1-j-k_1} P_y^{i+1-j-k_1} (1 - P_y)^{2k_1-i-1+j} \right\}$$

for  $i, j \geq m$  (5)

- For  $i \geq m$  and  $j < m$ , all servers are busy at the time of first arrival. At the time of second arrival, some servers are idle and the queue becomes empty. The one-step transition probability for this case is

$$p_{ij} = \sum_{k_1=\max(\lceil(i+1-j)/2\rceil,m-j)}^{\min(i+1-j,m)} \left\{ \binom{m}{k_1} P_x^{k_1} (1 - P_x)^{m-k_1} \cdot \binom{k_1}{i+1-j-k_1} P_y^{i+1-j-k_1} (1 - P_y)^{2k_1-i-1+j} \right\}$$

for  $i \geq m, j < m$  (6)

- As we assume that no more than two tasks can depart from any server between two successive task arrivals, we will also have the following one-step transition probabilities

$$p_{ij} = 0 \quad \text{for } (i + j - 1)/2 > m \tag{7}$$

### 3.3 Important parameters for task management

The following parameters should be calculated to support the decision making of task management: queue length information in the steady-state, queue length information after n-transition, blocking probability in the steady-state, blocking probability after n-transition, service waiting time in the steady-state and service waiting time after n-transition. Firstly, let  $\pi_k = \lim_{n \rightarrow \infty} \text{prob}[q_n = k], 0 \leq k \leq m + r$ , denote the probability of having queue length  $i$  in the steady-state solution. To obtain these values, the following equations are formed:

$$\pi_i = \sum_{j=0}^{m+r} \pi_j p_{ij}, \quad 0 \leq i \leq m+r \tag{8}$$

$$\sum_{i=0}^{m+r} \pi_i = 1 \tag{9}$$

Since the number of equations is  $m+r+2$  and the number of variables is  $m+r+1$ , one equation will be discarded while solving the equations. Our suggestion is to drop  $\pi_0 = \sum_{j=0}^{m+r} \pi_j p_{0j}$  for  $\frac{\lambda}{m\mu} \geq 1$ , and drop  $\pi_{m+r} = \sum_{j=0}^{m+r} \pi_j p_{mj}$  for  $\frac{\lambda}{m\mu} < 1$ . It has been proved in [23] that the above equations have a unique steady-state solution due to the fact that the corresponding Markov chain is ergodic. Based on the steady-state solution, we now have information about queue length and be able to get other important parameters. When the queue is full, it will block new arrivals. Consequently, the blocking probability in the steady-state equals to the probability of having  $m+r$  tasks in the system, i.e.,  $\pi_{m+r}$ . The LST of service waiting time in the steady-state can be calculated as  $W^*(s) = \sum_{k=0}^{m-1} \pi_k + \sum_{k=m}^{m+r} \pi_k (1 - s/\lambda e)^{k-m}$  [39]. Let  $W(x)$  be the CDF of service waiting time, the mean service waiting time  $\bar{W}$  is described by  $\bar{W} = \int_{x=0}^{\infty} (1 - W(x)) dx$ .

Secondly, we can use the one-step transition probability matrix to predict future states of queuing system. The  $n$ -step transition probability matrix  $P^{(n)}$  can be calculated as  $P^{(n)} = \prod_{k=1}^n P$ . The marginal distribution  $prob(q_n = j)$  is described by  $prob(q_n = j) = \sum_{k=0}^{m+r} p_{kj}^{(n)} prob(q_0 = k)$ . Given a certain starting state  $q_0 = i$ , we can get  $prob(q_n = j | q_0 = i) = p_{ij}^{(n)}$  as the queue length information. The blocking probability after  $n$ -step transition is  $p_{in}^{(n)}$  where  $\eta = m+r$ . The LST of service waiting time after  $n$ -step transition is  $W^*(s)^{(n)} = \sum_{k=0}^{m-1} p_{ik}^{(n)} + \sum_{k=m}^{m+r} p_{ik}^{(n)} (1 - s/\lambda e)^{k-m}$ . Using the corresponding CDF  $W(x)^{(n)}$ , the mean service waiting time after  $n$ -step transition is described by  $\bar{W}^{(n)} = \int_{x=0}^{\infty} W(x)^{(n)} x dx$ . The mean elapsed time of  $n$ -step transition  $\bar{E}^{(n)}$  can be calculated as  $\bar{E}^{(n)} = \frac{n}{\lambda}$ .

### 3.4 Algorithm design for task management

In this subsection, we describe a design of efficient online task management algorithm. The core concept of the algorithm is as follows: the task manager of each job queue observes the mean arrival rate during the past time slot and the current queue length. Based on the requirement of the corresponding multimedia service, the current utilization of servers and the queue dynamics, the task manager decides to request more VMs for QoS guarantee, or to stop running VMs for cost reduction.

The task manager first converts the QoS requirements of multimedia services to the constraints on one or many of the following parameters: queue length information, blocking probability and service waiting time. For example, the constraint on service waiting time must be very strict for real-time multimedia services. Next, the task manager defines a specific queue length as an indicator for judging the situation of server under-utilization.

The proposed online task management algorithm is invoked at the beginning of every time slot  $t$ . Since it is hard to predict future workload, the task manager uses the arrival rate of past time slot to predict queue dynamics in future time. The task manager checks current status of queue and potential risk of QoS violation within a certain period, which is a multiple of  $t$ . If a risk has been addressed, new VM requests are generated to increase the service rate. By adopting the allocation deadline in the VM requests, the VM allocator can achieve a long-term cost reduction in the next stage, while satisfying the QoS constraints for multimedia services. The under utilization of server is responded immediately by reducing the number of rented VMs. The formal description of task management algorithm is shown in Algorithm 1.

---

**Algorithm 1:** Efficient Task Management Algorithm
 

---

```

if is_new_timeslot() then
  if is_not_overloaded_now() then
    compute  $\pi_k()$  for  $0 \leq k \leq m + r$ ;
    compute  $\bar{W}()$ ;
    if is_not_overloaded_steady-state( $\pi_k, \bar{W}$ ) then
      if is_under_utilization_now() then
        while is_under_utilization_steady-state( $\pi_k, \bar{W}$ ) do
           $m --$ ;
          compute  $\pi_k()$  for  $0 \leq k \leq m + r$ ;
          compute  $\bar{W}()$ ;
        end
        cancel_previous_requests();
        reduce_number_of_server_to( $m$ );
      end
      return
    end
  else
     $n = 0$ ;
    compute  $p_{ij}^{(n)}()$ ;
    compute  $\bar{W}^{(n)}()$ ;
    compute  $\bar{E}^{(n)}()$ ;
    while is_not_overloaded_after_n_step( $p_{ij}^{(n)}, \bar{W}^{(n)}$ ) AND
       $\bar{E}^{(n)} \leq \text{given\_threshold}$  do
       $n ++$ ;
      compute  $p_{ij}^{(n)}()$ ;
      compute  $\bar{W}^{(n)}()$ ;
      compute  $\bar{E}^{(n)}()$ ;
    end
    if  $\bar{E}^{(n)} > \text{given\_threshold}$  then
      cancel_previous_requests();
      return
    end
  end
end
while is_overloaded_steady-state( $\pi_k, \bar{W}$ ) do
   $m ++$ ;
  compute  $\pi_k()$  for  $0 \leq k \leq m + r$ ;
  compute  $\bar{W}()$ ;
end
cancel_previous_requests();
increase_number_of_server_to( $m$ );
end

```

---

### 3.5 Time complexity analysis for task management

In Algorithm 1, we have two functions which are computational intensive, i.e.,  $compute\_π_k()$  and  $compute\_p_{ij}^{(n)}()$ . To compute  $π_k$ , the one-step transition probability matrix  $P$  has to be generated first. According to Eqs. (4)–(7), the complexity of calculating  $P$  is  $O((m+r)^3)$ . After that, each  $π_k$  can be found by using a Gaussian Elimination to solve the linear equations. The time complexity for such step is also  $O((m+r)^3)$ . Thus, the overall time complexity of  $compute\_π_k()$  is  $O((m+r)^3)$ . On the other hand,  $compute\_p_{ij}^{(n)}()$  requires only  $O((m+r)^2)$  time to compute  $P^{(n)}$  from  $P^{(n-1)}$ . Thus, the time complexity of  $compute\_π_k()$  dominates that of  $compute\_p_{ij}^{(n)}()$ . As  $compute\_π_k()$  has been put in the loop, the overall time complexity of entire task management approach is  $O(m_{max}(m+r)^3)$  where  $m_{max}$  is the maximum number of VMs that can be possibly requested.

## 4 Proposed resource management

### 4.1 Physical servers and VM requests

The cloud resources consists of  $nh$  physical servers defined as  $H = \{h_1, h_2, \dots, h_{nh}\}$ . In order to describe a physical server  $h_i$  ( $1 \leq i \leq nh$ ) in general, we use  $c_i$ ,  $m_i$ ,  $s_i$  and  $b_i$  to represent its CPU processing capability (expressed in millions of instructions per second—MIPS), memory space (expressed in MB), storage space (expressed in MB) and network bandwidth (expressed in KB/s), respectively. The similar modeling method can be found in a widely used Cloud simulator: CloudSim [40]. At time  $t$ , let  $fc_i(t)$ ,  $fm_i(t)$ ,  $fs_i(t)$ , and  $fb_i(t)$  be the percentage of free CPU processing capability, memory space, storage space and network bandwidth, respectively.

At time  $t$ , denote the set of arrived VM requests by  $L(t) = \{l_1, l_2, \dots, l_{nl(t)}\}$  where  $nl(t)$  is the total number of arrived requests from time 0 to time  $t$ . For a request  $l_j$  ( $1 \leq j \leq nl(t)$ ), let  $wt_j$ ,  $adj$  and  $st_j$  be the waiting time, allocation deadline and service time of that request, respectively. The waiting time is counted from the request arrives until it is finally allocated on a physical machine. In some cases,  $st_j$  may not be known or predictable before the processing of  $l_j$  is accomplished. Let  $v_j$  be the virtual machine for request  $l_j$ .

As we mentioned before, we use a  $nh$ -by- $nt(t)$  matrix  $A(t)$  to represent the allocation results at time  $t$  where the elements are binary. For any  $a_{ij}(t)$ ,  $a_{ij}(t) = 1$  means that the virtual machine  $v_j$  has been allocated on the physical machine  $h_i$ , and vice versa. Let  $rc_{ij}$ ,  $rm_{ij}$ ,  $rs_{ij}$  and  $rb_{ij}$  be  $v_j$ 's resource requirements on  $h_i$  regarding CPU capability, memory space, storage space and network bandwidth, respectively and all in percentage form. We assume that the overhead of VM creation and maintenance is also included in the resource requirements, which means  $rc_{ij}$ ,  $rm_{ij}$ ,  $rs_{ij}$  and  $rb_{ij}$  are the overall resource requirements for running  $v_j$  on  $p_i$ . In practical design, the values of  $rc_{ij}$ ,  $rm_{ij}$ ,  $rs_{ij}$  and  $rb_{ij}$  are acquired from user-supplied information, experimental data, benchmarking, application profiling, queuing calculation or other techniques.

## 4.2 Optimization goal for resource management

The optimization goal we proposed in this study is to minimize the trade-off between average waiting time and long-term service cost for cloud operator. The explanation and definition of the selected optimization goal can be found in the following contents.

We consider that the allocation deadline is clearly specified in the QoS requirement of any task. However, an early allocation can improve the QoS of multimedia services since the queue length can be potentially stabilized. To this end, we define the average service waiting time  $\overline{WT}$  as

$$\overline{WT} = \lim_{t \rightarrow \infty} \frac{1}{nl(t)} \sum_{j=1}^{nl(t)} wt_j \quad (10)$$

On the other hand, the cost reduction is also a major concern while operating a cloud system. The electricity energy consumption in computing facilities incurs the major cost of cloud during run time [41]. Recent studies identified that energy consumption scales linearly with resource utilization [42] and number of running servers [14]. By putting idle servers into sleep/power-saving mode, cloud provider can reduce idle power draw. Consequently, we assume that multimedia cloud provider will follow the same way to save cost. To that end, we use the number of running servers to represent the cost of cloud provider. The similar definition can be found in [41] which also aims at minimizing the amount of active components for achieving cost reduction. The other cost that incurs is VM fetching cost. To transfer, load and start VM instances, energy and resources will be consumed on server. However, this cost is usually undertaken by application providers rather than cloud provider. Thus, we do not put it into the definition of cloud provider's cost. To be more specific, the long-term cost can be defined as the cumulative running time of all active physical servers if we assume the servers' energy consumption is homogeneous. Let  $y_i(t)$  be the binary variable indicating whether a physical server  $h_i$  is active at time  $t$  ( $y_i(t) = 1$ ) or not ( $y_i(t) = 0$ ). The long term cost  $\bar{C}$  is then expressed as

$$\bar{C} = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{i=1}^{nh} \sum_{\tau=0}^{t-1} y_i(\tau) \quad (11)$$

Unfortunately, it is not always possible to optimize the average service waiting time  $\overline{WT}$  and the long-term service cost  $\bar{C}$  at the same time. Due to the dynamic and unpredictable features of multimedia service, cloud operator may receive many CPU intensive VM request during a short period. To minimize  $\overline{WT}$ , the cloud operator may perform immediate allocation after it receives any task. As a result, the CPU capability of the active physical servers can be fully utilized during the upcoming time slots while other resources are wasted more or less. Similar resource wasting situations can be found when the tasks arriving in a short period are extremely memory intensive, storage intensive or bandwidth intensive on average. Thus, we set up the optimization goal as to find an on-line allocation method that solves the following problem:

$$\begin{aligned} & \min_{A(t), t=0,1,2,\dots} \overline{WT} + K\bar{C} \\ & = \lim_{t \rightarrow \infty} \left( \frac{1}{nl(t)} \sum_{j=1}^{nl(t)} wt_j + K \frac{1}{t} \sum_{i=1}^{nh} \sum_{\tau=0}^{t-1} y_i(\tau) \right) \end{aligned} \tag{12}$$

$$s.t., \quad fc_i(t), fm_i(t), fs_i(t), fb_i(t) \geq 0 \quad \forall t, i \tag{13}$$

$$\begin{aligned} & \exists p_i : \quad \begin{aligned} & fc_i(t) \geq rc_{ij} \\ & fm_i(t) \geq rm_{ij} \\ & fs_i(t) \geq rs_{ij} \\ & fb_i(t) \geq rb_{ij} \end{aligned} \quad \forall t, j \end{aligned} \tag{14}$$

$$t \leq adj|j, \sum_{i=1}^{nh} a_{ij}(t) = 0 \quad \forall t \tag{15}$$

The parameter  $K$  in (12) is a non-negative weight that is chosen as desired to affect a tradeoff between average waiting time and long-term cost. Constraints (13) and (14) guarantee the resource sufficiency of single server and entire cloud, respectively. Equation (15) shows that all VMs must be allocated before their allocation deadline.

The above problem can be mapped to the multi-dimensional bin-packing problem at each time slot. The goal of this problem is to map several items, where each item represents a tuple containing its dimensions, into the smallest number of bins as possible. We consider each virtual machine request as an item and the dimensions as its CPU, memory, storage and bandwidth requirements, and the goal is to minimize the number of physical servers that must be used to place all virtual machines, respecting physical servers corresponding capacities. As the multi-dimensional bin-packing problem is a well-known NP-complete problem, it is proved that the allocation problem in our scenario is also a NP-complete problem at any time  $t$ . Thus, it is only feasible to use fast algorithm, such as heuristic, to solve the allocation problem.

### 4.3 Key parameters for resource management

We define the following parameters to represent the overall resource utilization condition of any single physical server. Given a physical server  $h_i$ , The first parameter is the mean of resource usage  $\mu_i(t)$  at time  $t$ .

$$\mu_i(t) = \frac{fc_i(t) + fm_i(t) + fs_i(t) + fb_i(t)}{4} \tag{16}$$

$\mu_i(t)$  provides a direct view showing how efficiently the resources of  $h_i$  are utilized at time  $t$ . However, there is another parameter which indicates the overall resource utilization condition implicitly. We use  $\sigma_i(t)$  to denote the situation of resource balance.

$$\begin{aligned}
temp_1 &= (\mu_i(t) - fc_i(t))^2 \\
temp_2 &= (\mu_i(t) - fm_i(t))^2 \\
temp_3 &= (\mu_i(t) - fs_i(t))^2 \\
temp_4 &= (\mu_i(t) - fb_i(t))^2 \\
\sigma_i(t) &= \frac{\sqrt{temp_1+temp_2+temp_3+temp_4}}{2}
\end{aligned} \tag{17}$$

It can be seen intuitively that a better overall resource utilization will occur when  $\mu_i(t)$  increases and  $\sigma_i(t)$  decreases. Thus, we combine these two parameters into a single parameter  $cv_i(t)$ . The definition of  $cv_i(t)$  is

$$cv_i(t) = \frac{\sigma_i(t)}{\mu_i(t)} \tag{18}$$

For any unallocated request  $l_j$  at time  $t$ , the waiting time  $wt_j$  and the allocation deadline  $ad_j$  are combined into one parameter  $pr_j(t)$  to provide the optimization concern of average service waiting time.

$$pr_j(t) = g\left(\frac{wt_j}{ad_j - t + wt_j}\right) \tag{19}$$

$$g(x_1) < g(x_2) | x_1 > x_2 \tag{20}$$

$$\lim_{x \rightarrow 0} g(x) = 1, \lim_{x \rightarrow 1} g(x) = 0 \tag{21}$$

where  $g(x)$  is specified by cloud operator to indicate the relative importance of waiting time over resource utilization. Constraint (20) guarantees that  $pr_j(t)$  decreases when the time approaches to the allocation deadline of  $l_j$ . The range of  $pr_j(t)$  is limited in (21) as  $[0, 1]$ .

#### 4.4 Online allocation for resource management

We modify four existing heuristic algorithms by changing the metric and adding a threshold value to achieve dynamic control. The modified heuristics are user-directed assignment (UDA) [43], Min–Min [43], Max–Min [43] and sufferage heuristic [44]. A UDA heuristic assigns each VM, in arbitrary order, to the physical machine with the best trade-off metric value. A Min–Min heuristic selects the VM/server pair that produces the overall minimum metric value. In Max–Min, the minimum value of the metric is selected for each VM first. Among all VMs, the one that produces the overall maximum metric value is finally chosen for allocation. A Sufferage heuristic attempts to allocate a task that would “suffer” most in terms of expected metric value if that particular server is not assigned to it. The metric in all heuristics are defined as a combination of  $cv_i(t)$  and  $pr_j(t)$  by taking a multiplication, i.e.,  $cv_i(t) \times pr_j(t)$ . According to (20) and (21),  $cv_i(t) \times pr_j(t)$  equals to  $cv_i(t)$  when task  $l_j$  just arrives. In this case, the value of metric is fully based on the overall resource utilization of  $p_i$  after allocating  $v_j$  to that physical server. The value of  $cv_i(t) \times pr_j(t)$  goes down to 0 when the time reaches the allocation deadline of  $l_j$ . We also introduce a dynamic threshold

value  $\xi(t)$ . If a candidate allocation  $a_{ij}(t) = 1$  satisfies  $cv_i(t) \times pr_j(t) \leq \xi(t)$ , it is considered as an approved allocation. At time  $t$ , a request  $l_j$  can not be allocated when there does not exist any physical server  $h_i$  satisfies  $cv_i(t) \times pr_j(t) \leq \xi(t)$ . We provide the detailed steps of allocation process as follows.

*Step 1:* The allocation process starts at time  $t = 0$ . The initial threshold value is defined as  $\xi(0) = +\infty$ , which means the threshold is not adopted. For any physical machine  $h_i$ , we have  $fc_i(0) = fm_i(0) = fs_i(0) = fb_i(0) = 100\%$ . The task set is an empty set, i.e.,  $L(0) = \emptyset$ . The initial selected heuristic is Min–Min. In fact, the initial selection of threshold value and heuristic is not an important issue. Both of them will be changed at the end of first long-period  $T = 1$ .

*Step 2:* During a short time slot  $t$ , the urgent allocation requests may arrive. To handle them, we use the selected heuristic and remove the constraint of threshold from it. As each urgent request is executed individually and immediately, the actual metric is  $cv_i(t)$  only, regardless of the heuristics.

*Step 3:* At the end of a short time slot  $t$ , the regular allocation requests need to be handled. One of the heuristics presented before is used. The method of heuristic selection can be found in Step 1 and Step 5.

*Step 4:* Repeat Step 2 and Step 3 until the end of a long-period  $T$ . The length of  $T$  should be long enough to show the overall trend of workload fluctuation. If the time is also the end of a short time slot, finish Step 3 before starting this step. According to the pre-defined conditions, choose the VMs that need to be migrated first. Try to allocate the VMs use one of the heuristics presented before, and check whether the condition is eliminated or not. If so, then execute the allocation.

*Step 5:* At this step, the threshold value and heuristic for next long-period are determined. The method is to adopt different threshold values in each heuristic to find the offline allocation results for the past period. We use (12) to judge which is the optimal threshold value and which is the optimal heuristic. The optimal heuristic is directly applied in the next long-period. The optimal threshold value passes PID controller which generates the threshold value that can be used in next long-period.

Before the first long-period  $T$  starts, an initial threshold value  $\xi(0) = +\infty$  is applied. By the end of period  $T$ , the optimal threshold value decided by offline allocation is denoted as  $\xi_o(T)$ . If  $T = 1$ , adopt  $\xi_o(T)$  in the next long-period. Otherwise, let  $\xi_e(T) = \xi_o(T) - \xi(T)$  be the error of threshold value during period  $T$ . Our target is to eliminate the error for following period. Thus, we adopt the general method of PID controller and describe the formula for calculating the dynamic threshold as follows:

$$\xi(T + 1) = \xi(T) + u(T) \tag{22}$$

$$u(T) = k_p \times \xi_e(T) + k_i \times \sum_{j=1}^T \xi_e(j) + k_d \times (\xi_e(T) - \xi_e(T - 1)) \tag{23}$$

where  $k_p$ ,  $k_i$  and  $k_d$  are the weighting parameters (tunable).

The formula (23) involves three separate parts: the proportional, the integral and derivative values, denoted as  $\xi_e(T)$ ,  $\sum_{j=1}^T \xi_e(j)$ , and  $(\xi_e(T) - \xi_e(T - 1))$ , respectively. These values can be interpreted in terms of time:  $\xi_e(T)$  depends on the present error,  $\sum_{j=1}^T \xi_e(j)$  on the accumulation of past errors, and  $(\xi_e(T) - \xi_e(T - 1))$  is a



prediction of future errors, based on current rate of change. The weighted sum of these three parameters  $u(t)$  is used to adjust the current threshold value  $\xi(T)$ .  $k_p$ ,  $k_i$  and  $k_d$  are tunable parameters. By tuning the three parameters, the PID controller can provide control action designed for specific process requirements. The readers who are interested in the tuning algorithms may refer to [45] for more details.

#### 4.5 System stability

In the proposed system, the stability can be measured by evaluating the states of queue length which are related to many QoS factors, such as waiting time and response time. Define  $Q_i(t)$  as the length of the  $i$ th queue at time slot  $t$ . Following quadratic Lyapunov function is used to represent the state of whole system at time slot  $t$

$$L(t) = \frac{1}{2} \sum_{i=1}^N Q_i(t)^2 \quad (24)$$

For a single queue, the Lyapunov drift  $\Delta_i(t)$  is denoted as

$$\Delta_i(t) = \frac{1}{2} Q_i(t+1)^2 - \frac{1}{2} Q_i(t)^2 \quad (25)$$

And the Lyapunov drift of whole system  $\Delta(t)$  is defined as

$$\Delta(t) = L(t+1) - L(t) \quad (26)$$

Since the QoS of multimedia services are guaranteed by the task manager, it is unlikely to observe blocking event where the corresponding queue length is  $m+r$ . Thus, the change of a single queue follows

$$Q_i(t+1) = \max[Q_i(t) + \lambda_i(t) - m_i(t)\mu_i, 0] \quad (27)$$

where  $\lambda_i(t)$ ,  $m_i(t)$  and  $\mu_i$  are arrival rate during  $t$ , number of VMs during  $t$  and service rate of single VM. The quadratic form of above equation is bounded as

$$Q_i(t+1)^2 \leq (Q_i(t) + \lambda_i(t) - m_i(t)\mu_i)^2 \quad (28)$$

The Lyapunov drift is bounded as

$$\Delta_i(t) \leq \frac{1}{2}(\lambda_i(t) - m_i(t)\mu_i)^2 + Q_i(t)(\lambda_i(t) - m_i(t)\mu_i) \quad (29)$$

Now we define the conditional expected Lyapunov drift of a single queue  $i$  as

$$E[\Delta_i(t)|Q_i(t)] \leq \frac{1}{2}(\lambda_i(t) - m_i(t)\mu_i)^2 + Q_i(t)E[(\lambda_i(t) - m_i(t)\mu_i)|Q_i(t)] \quad (30)$$

And the conditional expected Lyapunov drift of whole system is

$$E[\Delta(t)|Q(t)] \leq \frac{1}{2} \sum_{i=1}^N (\lambda_i(t) - m_i(t)\mu_i)^2 + \sum_{i=1}^N Q_i(t)E[(\lambda_i(t) - m_i(t)\mu_i)|Q(t)] \quad (31)$$

In the task management stage, the stability of any queue can be controlled by adjusting the allocation deadline. By setting an earlier VM allocation deadline, the bounding of queue length in (30) becomes tighter due to the reduction of  $E[(\lambda_i(t) - m_i(t)\mu_i)|Q_i(t)]$ . As a result, the QoS of the corresponding multimedia service can be improved and stabilized.

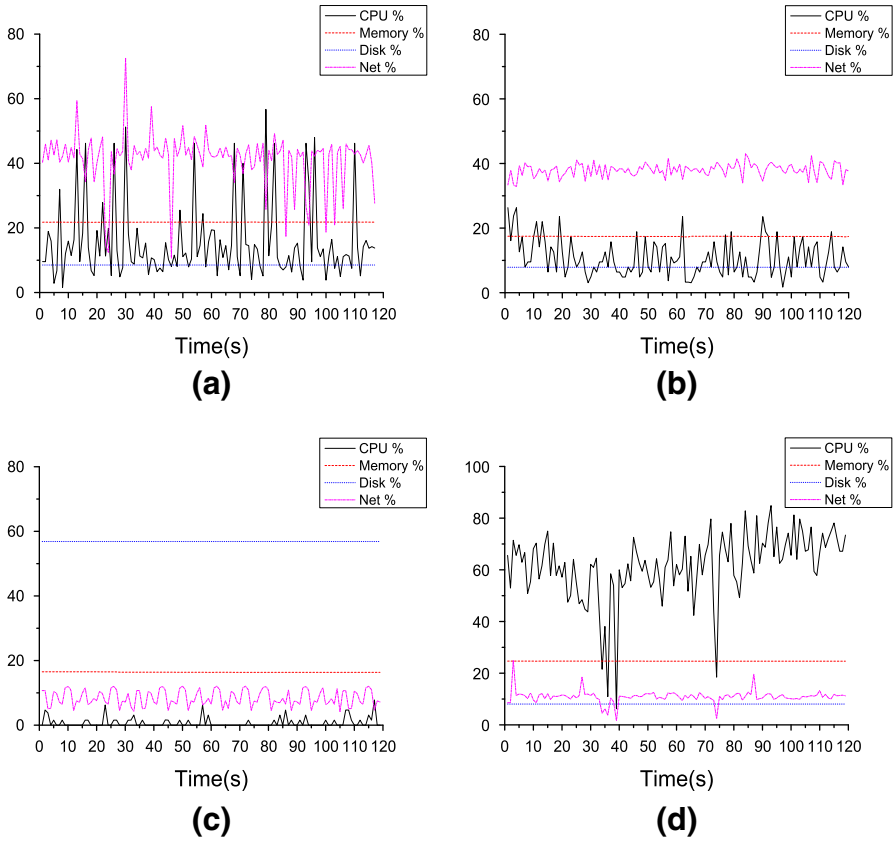
In the resource management stage, the stability of entire system can be also controlled by adjusting the allocation time, which is achieved by changing the function of  $g()$  in (19). Earlier VM allocations lead to the reduction of  $E[(\lambda_i(t) - m_i(t)\mu_i)|Q(t)]$  in (31) for every queue. Consequently, the QoS of the entire system can be improved and stabilized.

It is notable that earlier VM allocations bring cost increase in both stages. Since the arrival rate  $\lambda_i(t)$  may fluctuate over time, earlier VM allocations will cause unnecessary fetching of VMs due to the earlier grow of  $m_i(t)$  when the arrival rate only increases for a short period. Also, the possibility of finding an allocation with better  $cv_i(t)$  is reduced by giving more weight on the allocation waiting time. This will result in low server resource utilization and high operation cost for cloud operator. The detailed impact of deadline setting can be viewed in the simulation section.

## 5 Simulations

### 5.1 Setup

Understanding cloud-based multimedia workload is a challenging task. To simplify the task of workload selection and setup, we start from identifying some key classes of video operations and then carefully selecting their implementations. Specifically, we focus on four typical paradigms: video streaming and monitoring, face detection, video encoding/transcoding and video storage. To understand the characteristics of our multimedia workloads, we analyze their run-time statistics collected while running the applications on Amazon Elastic Computing Cloud (EC2). We rent a M1 Small VM having one Intel(R) Xeon(R) E5430 @2.66 GHz CPU unit, one CPU core, 1.7 GiB memory, 1 Gbps bandwidth and 30G hard drive with Microsoft Server 2008 Base 64-bit. We use the performance monitor of Windows to record the resource utilization of CPU, memory, storage and network bandwidth. Figure 2 illustrates the resource utilization rates of the aforementioned workloads over the course of their execution. We only plot partial utilization traces that represent the key execution phases. The y-axis represents total resource utilization. The CPU utilization, memory utilization, disk space utilization and network bandwidth in percentage form can be found in the

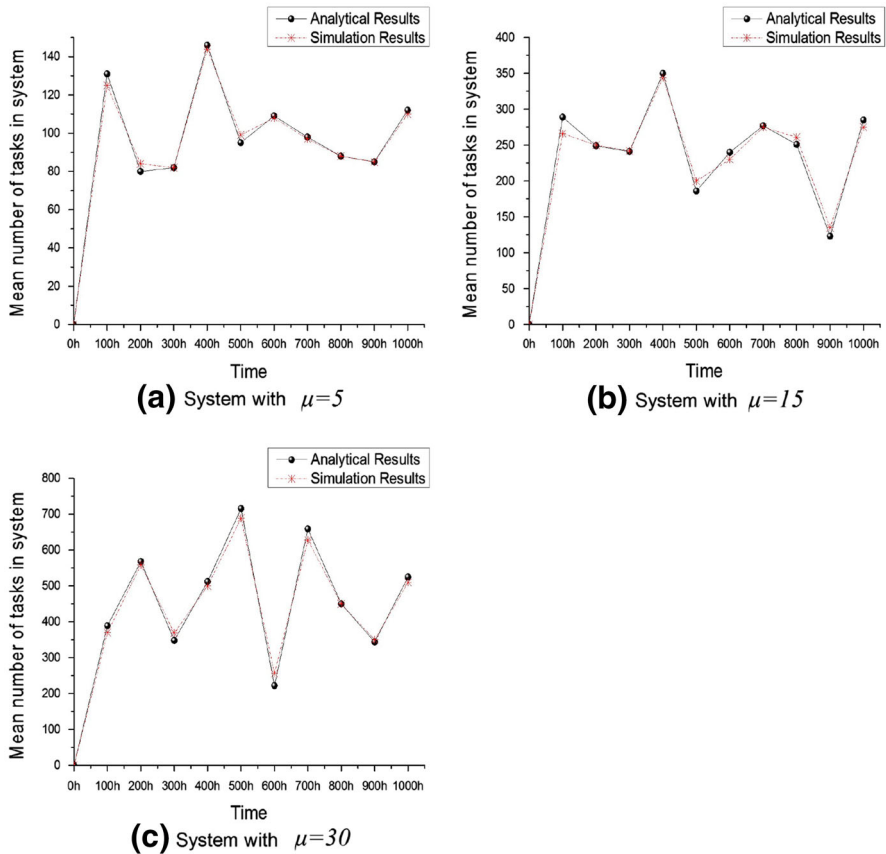


**Fig. 2** Workload of four typical multimedia services. **a** Face detection, **b** video streaming, **c** video storage and **d** video encoding

figure. As we can see, a significant variation exists in the resource utilization across several workloads.

To test the efficiency of our task management approach, we focused on the construction of workload simulator. We have considered a specific queue with three different settings including mean service time ( $\frac{1}{\mu} = 5, 15$  and  $30$ ), minimum waiting length ( $30, 50$  and  $100$ ), and maximum waiting length ( $r = 100, 250, 500$ ), respectively. When the waiting length of a queue is less than the minimum waiting length, the task manager starts to reduce the number of running VM. Meanwhile, the waiting length must be kept less than the maximum waiting length such that  $prob(q_i = m + r) < 0.1 \%$  for every  $i$ . In all cases, the arrival rate is fixed as  $\lambda = 30$ . The cloud operator is assumed to allocate all VM requests submitted from this queue at the time slots near to their allocation deadlines.

To test the efficiency of proposed resource management approach, we first generate 100 heterogeneous queues with different initial settings including service rate, minimum waiting length and maximum waiting length. The arrival rate of each queue is varied over time, which are changed every 100 time-slots. Those queues are assumed



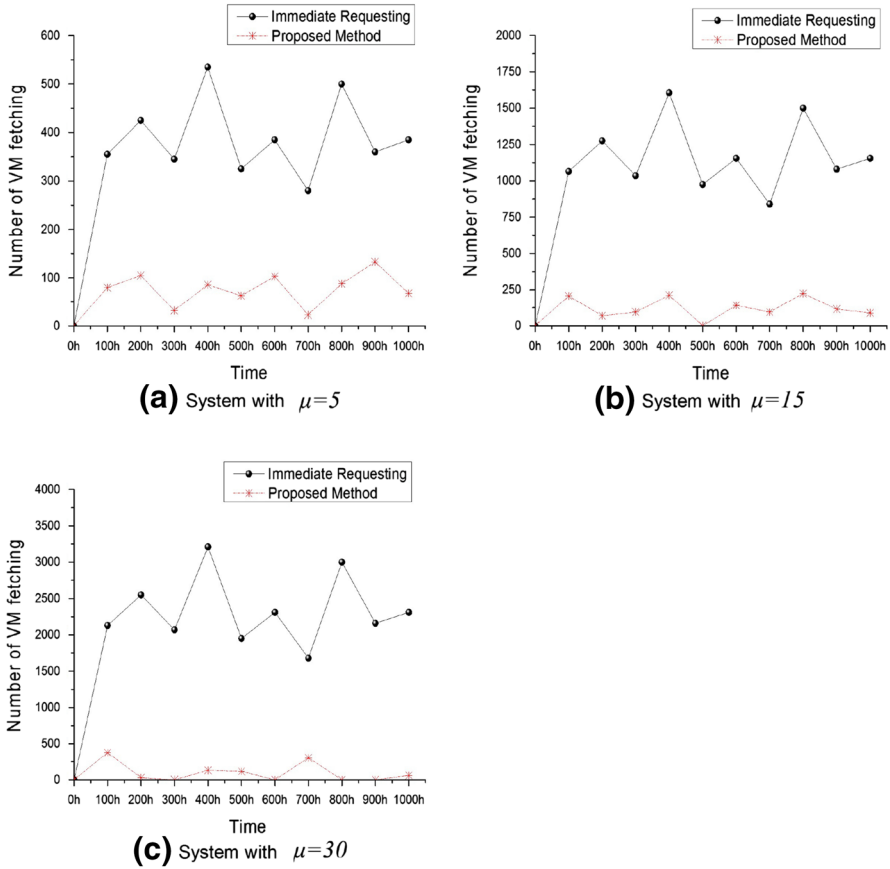
**Fig. 3** Analytical results vs. simulated results

to process the workload of video streaming/monitoring, face detection, video encoding/transcoding and video storage by submitting VM requests with heterogeneous resource requirements and allocation deadline. For cloud operator, the total duration of VM allocation request arrival is between time-slot 0 and 1,000. The VM resource requirement of each queue follows a normal distribution with mean ratios of CPU/memory, CPU/disk and CPU/net as shown in Fig. 2. The allocation waiting time of any VM allocation request is less than 1,000 time-slots.

## 5.2 Simulation results

### 5.2.1 Analytical vs. simulated

We first validate the effectiveness of proposed queuing and Markov prediction model. Figure 3 presents the comparison between analytical results and simulated results. The mean values of queue length are counted every 100 time = slots. From the figure we can see that the errors of our analytical model are very small in all cases. Besides,

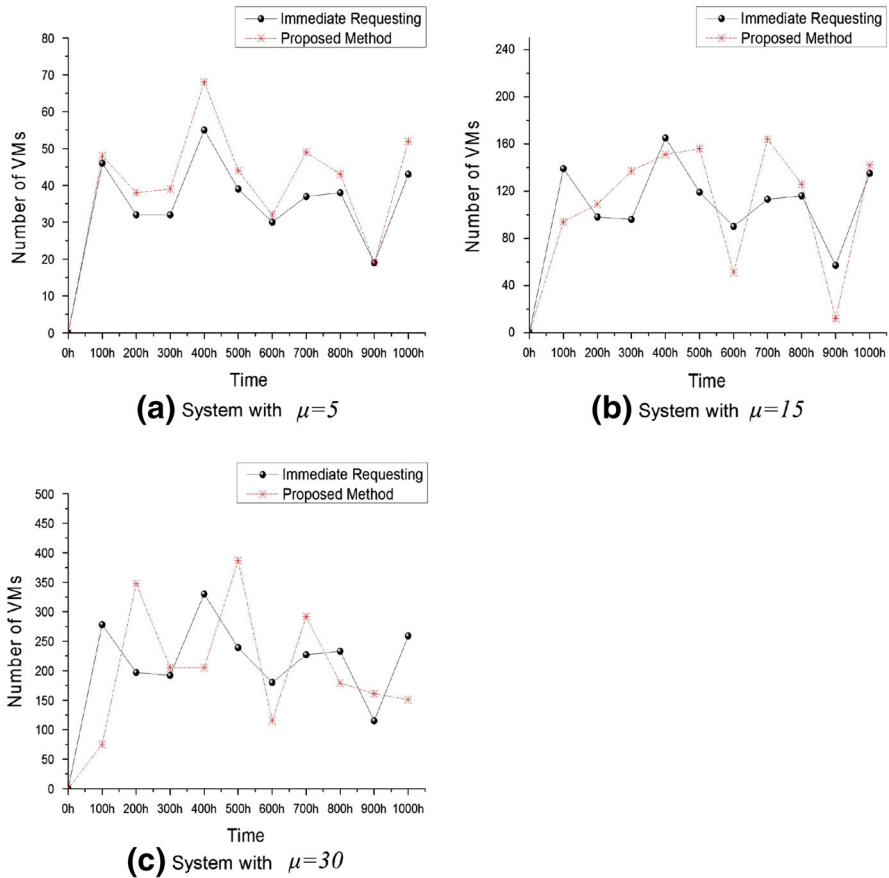


**Fig. 4** Number of allocated VMs

we do not observe underestimation of queue length when the workload is at peak. According to these results, we can conclude that the proposed analytical model can provide near-to-accurate prediction and will not cause unexpected QoS violation.

5.2.2 Cost optimization

In task management, we implemented two approaches: proposed approach and immediate response approach. The immediate response approach can be found in [11] which generates VM requests and stops running VMs based on the current task arrival rate and service rate. We count the number of allocated VMs in every 100 time-slots and present the results in Fig. 4. The results show that the proposed task management approach significantly reduces the number of VM fetchings since the use of allocation deadline effectively avoids unnecessary VM allocation. From Fig. 5 we can see that the average number of rented VMs by proposed approach is actually similar to number of running VMs that can be found in the immediate response approach. These results proves that our approach effectively avoids the following situation from happening:



**Fig. 5** Number of running VMs

the sudden increase of arrival rate during a few time slots cause many VM requests generated; cloud provider has to fetch the corresponding number of VMs; these VMs instances are stopped again very soon since the arrival rate decreases back to the original level immediately; to transfer, load and start VM instances, energy and resources has been consumed on server. Overall the cumulative number of running instances are only slightly differed from our approach which has allocation deadline settings; but the number of fetching raises up significantly. This type of workload pulse can be found in many event driven multimedia tasks, such as suspicious event detection and analytic for video surveillance. It can be seen that the advantage of our proposed approach becomes larger as the mean service time increases.

In resource management, we implemented four approaches: FCFS, load balancing [12], server consolidation [46], and proposed method. We first present the results of total cost in terms of cumulative machine hours. As we can see from Fig. 6, FCFS and load balancing have the worst performance among five approaches. The server consolidation approach, since it considers overall resource utilization, provides better results comparing with FCFS and load balancing. By using the proposed metric of

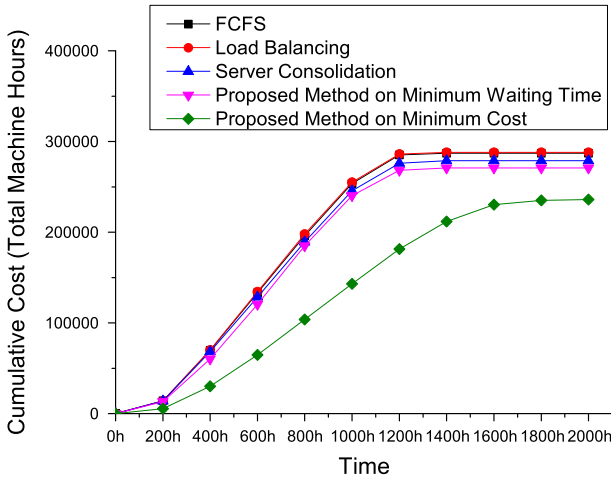


Fig. 6 Total cost on cloud simulator

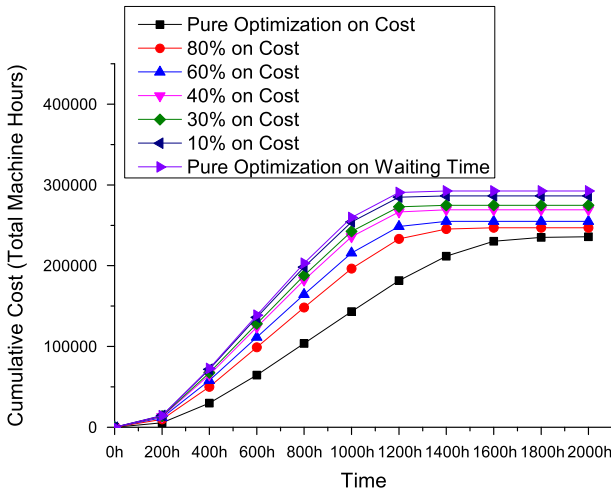
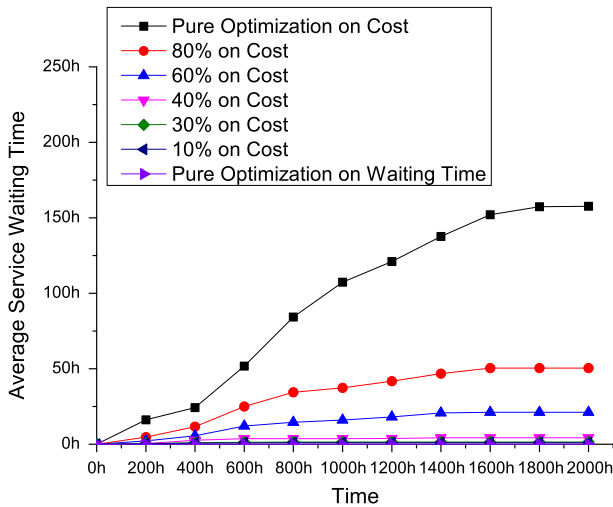


Fig. 7 Total cost in different tradeoff settings

our proposed method, the efficiency of VM allocation is increased. However, the improvement is not magnificent when waiting time is chosen as the optimization goal. Since all VMs are allocated immediately to minimize the waiting time, the proposed dynamic threshold does not affect the results of allocation. Thus, the best approach regarding cost optimization appears when we use both proposed metric and dynamic threshold where more than 20 % of the cost can be saved.

5.2.3 Tradeoff between cost and waiting time

In Figs. 7 and 8, we present the tradeoff between long term cost and average waiting time that has been achieved by using proposed VM allocation approach. The function



**Fig. 8** Average task waiting time in different tradeoff settings

$g(x)$  was adjusted to achieve different tradeoff during the allocation process. On the one hand, Fig. 7 indicates that the total cost increases as we put smaller weight on cost optimization. On the other hand, the average waiting time in Fig. 8 decreases while increasing the cost. The differences among several optimization settings are obvious. When the optimization on cost is 90 %, the average waiting time is more than 50 time slots in our simulation. However, this value is reduced to less than 5 time slots if we set the optimization on cost as 40 %.

## 6 Conclusion

This paper presents a two-stage approach: media task management and cloud resource management for multimedia services in a cloud computing environment. The concept of allocation deadline is introduced in both of these approaches which makes the proposed solution unique from existing methods and benefits both multimedia service provider and cloud operator. For the media task management, a queuing based approach is proposed. In this approach, an online and efficient task management algorithm using Markov analysis and prediction was developed. For the resource management, an efficient heuristic algorithm was presented that can select and allocate task in an active and dynamic way through a dynamic controller. Various simulations were carried out to evaluate the performance of the proposed approaches as compared to the existing methods. The results showed that the proposed solution provided cost-effective and flexible task and resource management than the state-of-the-art approaches. However, in this work we did not include quality of experience (QoE), media play-back quality, media service profiling and benchmarking. As for the future works, we would incorporate some of above as a part of the future work. We believe that our proposed allocation approach can adapt such settings. Although our method



solves the problem in a polynomial time, we still plan to further reduce the time complexity of our method to make it more scalable for the scenario where the number of servers increases greatly.

**Acknowledgments** The authors would like to extend their sincere appreciation to the Deanship of Scientific Research at King Saud University for its funding of this research through the Research Group Project no RGP-VPP-258.

## References

1. Zhu W, Luo C, Wang J, Li S (2011) Multimedia cloud computing. *IEEE Signal Process Mag* 28(3):59–69. doi:[10.1109/MSP.2011.940269](https://doi.org/10.1109/MSP.2011.940269)
2. Amreen K, Kamal K (2011) Mobile cloud computing as a future of mobile multimedia database. *Int J Comput Sci Commun* 2:29–221
3. Dey S (2012) Cloud mobile media: opportunities, challenges, and directions. In: 2012 international conference on computing, networking and communications (ICNC), pp 929–933
4. Kumar K, Lu YH (2010) cloud computing for mobile users: can offloading computation save energy? *Computer* 43(4):51–56. doi:[10.1109/MC.2010.98](https://doi.org/10.1109/MC.2010.98)
5. Chen YL, Chen TS, Huang TW, Yin LC, Wang SY, Chiueh T (2013) Intelligent urban video surveillance system for automatic vehicle detection and tracking in clouds. In: 2013 IEEE 27th international conference on advanced information networking and applications (AINA), pp 814–821. doi:[10.1109/AINA.2013.23](https://doi.org/10.1109/AINA.2013.23)
6. Lin CF, Yuan SM, Leu MC, Tsai CT (2012) A framework for scalable cloud video recorder system in surveillance environment. In: 9th international conference on ubiquitous intelligence and computing and 9th international conference on autonomic and trusted computing (UIC/ATC), pp 655–660
7. Miao D, Zhu W, Luo C, Chen CW (2011) Resource allocation for cloud-based free viewpoint video rendering for mobile phones. In: Proceedings of the 19th ACM international conference on multimedia, MM '11, pp 1237–1240. ACM, New York. doi:[10.1145/2072298.2071983](https://doi.org/10.1145/2072298.2071983)
8. Yi S, Jing X, Zhu J, Zhu J, Cheng H (2012) The model of face recognition in video surveillance based on cloud computing. In: *Advances in computer science and information engineering*, pp 105–111. Springer, New York
9. Nan X, He Y, Guan L (2011) Optimal resource allocation for multimedia cloud based on queuing model. In: 2011 IEEE 13th international workshop on multimedia signal processing (MMSP), pp 1–6. doi:[10.1109/MMSP.2011.6093813](https://doi.org/10.1109/MMSP.2011.6093813)
10. Nan X, He Y, Guan L (2012) Optimal allocation of virtual machines for cloud-based multimedia applications. In: 2012 IEEE 14th international workshop on multimedia signal processing (MMSP), pp 175–180. doi:[10.1109/MMSP.2012.6343436](https://doi.org/10.1109/MMSP.2012.6343436)
11. Nan X, He Y, Guan L (2012) Optimal resource allocation for multimedia cloud in priority service scheme. In: 2012 IEEE international symposium on circuits and systems (ISCAS), pp 1111–1114
12. Wen H, Hai-ying Z, Chuang L, Yang Y (2011) Effective load balancing for cloud-based multimedia system. In: 2011 international conference on electronic and mechanical engineering and information technology (EMEIT), vol 1, pp 165–168. doi:[10.1109/EMEIT.2011.6022888](https://doi.org/10.1109/EMEIT.2011.6022888)
13. Aisopos F, Tserpes K, Varvarigou T (2011) Resource management in software as a service using the knapsack problem model. *Int J Prod Econ* 141(2):465–477. doi:[10.1016/j.ijpe.2011.12.011](https://doi.org/10.1016/j.ijpe.2011.12.011). <http://www.sciencedirect.com/science/article/pii/S0925527311005275>
14. Beloglazov A, Abawajy J, Buyya R (2011) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Futur Gener Comput Syst*. doi:[10.1016/j.future.2011.04.017](https://doi.org/10.1016/j.future.2011.04.017). <http://www.sciencedirect.com/science/article/pii/S0167739X11000689>
15. Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr Comput Pract Exp* 24:1397–1420
16. Goiri Í, Berral JL, Fitó JO, Julià F, Nou R, Guitart J, Gavaldà R, Torres J (2012) Energy-efficient and multifaceted resource management for profit-driven virtualized data centers. *Futur Gener Comput Syst* 28(5):718–731. doi:[10.1016/j.future.2011.12.002](https://doi.org/10.1016/j.future.2011.12.002). <http://www.sciencedirect.com/science/article/pii/S0167739X11002366>

17. Hassan MM, Hossain MS, Sarkar AMJ, Huh E-N (2012) Cooperative game-based distributed resource allocation in horizontal dynamic cloud federation platform. *Inf Syst Front* 1–20. doi:[10.1007/s10796-012-9357-x](https://doi.org/10.1007/s10796-012-9357-x)
18. Lin W, Qi D (2010) Research on resource self-organizing model for cloud computing. In: 2010 international conference on internet technology and applications, pp 1–5. doi:[10.1109/ITAPP.2010.5566394](https://doi.org/10.1109/ITAPP.2010.5566394)
19. Nguyen Van H, Dang Tran F, Menaud JM (2009) Autonomic virtual resource management for service hosting platforms. In: Proceedings of the 2009 ICSE workshop on software engineering challenges of cloud computing, CLOUD '09, pp 1–8. IEEE Computer Society, Washington. doi:[10.1109/CLOUD.2009.5071526](https://doi.org/10.1109/CLOUD.2009.5071526)
20. Stillwell M, Schanzenbach D, Vivien F, Casanova H (2010) Resource allocation algorithms for virtualized service hosting platforms. *J Parallel Distrib Comput* 70(9):962–974. doi:[10.1016/j.jpdc.2010.05.006](https://doi.org/10.1016/j.jpdc.2010.05.006). <http://www.sciencedirect.com/science/article/pii/S0743731510000997>
21. Teng F, Magoule SF (2010) Resource pricing and equilibrium allocation policy in cloud computing. In: 2010 IEEE 10th international conference on computer and information technology (CIT), pp 195–202. doi:[10.1109/CIT.2010.70](https://doi.org/10.1109/CIT.2010.70)
22. Wei G, Vasilakos AV, Yao Z, Xiong N (2010) A game-theoretic method of fair resource allocation for cloud computing services. *J Supercomput* 54:252–269
23. Khazaei H, Mišić J, Mišić VB (2012) Performance analysis of cloud computing centers. In: Quality, reliability, security and robustness in heterogeneous networks, pp 251–264. Springer, New York
24. Ranjan R, Buyya R, Harwood A (2005) A case for cooperative and incentive-based coupling of distributed clusters. In: IEEE international conference on cluster computing, pp 1–11
25. Rahman M, Ranjan R, Buyya R, Benatallah B (2011) A taxonomy and survey on autonomic management of applications in grid computing environments. *Concurr Comput Pract Exp* 23(16):1990–2019
26. Wang L, Tao J, Ranjan R, Marten H, Streit A, Chen J, Chen D (2013) G-Hadoop: MapReduce across distributed data centers for data-intensive computing. *Futur Gener Comput Syst* 29(3):739–750
27. Paul AK, Park JS (2013) Multiclass object recognition using smart phone and cloud computing for augmented reality and video surveillance applications. In: 2013 international conference on informatics, electronics and vision (ICIEV), pp 1–6
28. Hossain MS, Hassan MM, Qurishi MA, Alghamdi A (2012) Resource allocation for service composition in cloud-based video surveillance platform. In: IEEE international conference on multimedia and expo workshops (ICMEW), pp 408–412
29. Hassan MM (2014) Cost-effective resource provisioning for multimedia cloud-based e-health systems. *Multimed Tools Appl*. doi:[10.1007/s11042-014-2040-0](https://doi.org/10.1007/s11042-014-2040-0)
30. Saini M, Wang X, Atrey PK, Kankanhalli M (2012) Adaptive workload equalization in multi-camera surveillance systems. *IEEE Trans Multimed* 14(3):555–562
31. Yang B, Feng T, Yuan-Shun D, Suchang G (2009) Performance evaluation of cloud service considering fault recovery. In: Cloud computing, pp 571–576. Springer, Berlin
32. Ma Bobby NW, Mark JW (1995) Approximation of the mean queue length of an M/G/c queueing system. *Oper Res* 43(1): 158–165
33. Smith J (2003) MacGregor.: *M/G/c/K* blocking probability models and system performance. *Perform Eval* 52(4):237–267
34. Wang L, Khan SU, Chen D, Koodziej J, Ranjan R, Xu CZ, Zomaya A (2013) Energy-aware parallel task scheduling in a cluster. *Futur Gener Comput Syst* 29(7):1661–1670
35. Heyman DP, Sobel MJ (2003) Stochastic models in operations research: stochastic optimizations, vol 2. <http://DoverPublications.com>
36. Grimmett G, Stirzaker D (2001) Probability and random processes. Oxford University Press, Oxford
37. Hassan Sodhro A, Ye L (2013) Medical quality-of-service optimization in wireless telemedicine system using optimal smoothing algorithm. In: E-Health Telecomm Syst Netw 2:1
38. Doshi B (1986) Queueing systems with vacations a survey. *Queueing Syst* 1(1):29–66
39. Marshall KT, Wolff RW (1971) Customer average and time average queue lengths and waiting times. *J Appl Probab* 535–542
40. Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw Pract Exp* 41(1):23–50
41. Lee YC, Zomaya AY (2012) Energy efficient utilization of resources in cloud computing systems. *J Supercomput* 60(2):268–280

42. Meisner D, Gold BT, Wenisch TF (2009) PowerNap: eliminating server idle power. In: ACM Sigplan notices, vol 44(3), pp 205–216. ACM, New York
43. Freund RF, Gherrity M, Ambrosius S, Campbell M, Halderman M, Hensgen D, Keith E, Kidd T, Kussow M, Lima JD, et al (1998) Scheduling resources in multi-user, heterogeneous, computing environments with smartnet. In: Proceedings of the 7th heterogeneous computing workshop (HCW 98), pp 184–199
44. Maheswaran M, Ali S, Siegel HJ, Hensgen D, Freund RF (1999) Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. *J Parallel Distrib Comput* 59(2):107–131
45. Atherton DP (1999) PID controller tuning. *Comput Control Eng J* 10(2):44–50
46. Ferreto TC, Netto MAS, Calheiros RN, De Rose CAF (2011) Server consolidation with migration control for virtualized data centers. *Future Gener Comput Syst* 27:1027–1034