Joint Optimization of Resource Allocation and Workload Scheduling for Cloud based Multimedia Services

Xiaoming Nan, Yifeng He, Ling Guan

Department of Electrical and Computer Engineering
Ryerson University, Toronto, Ontario, Canada

Abstract—With the development of cloud technology, cloud computing has been increasingly used as distributed platforms for multimedia services. However there are two fundamental challenges for service providers: one is resource allocation, and the other is workload scheduling. Due to the rapidly varying workload and strict response time requirement, it is difficult to optimally allocate virtual machines (VMs) and assign workload. In this paper, we study the resource allocation and workload scheduling problem for cloud based multimedia services. Specifically, we introduce a queuing model to quantify the resource demands and service performance, and a directed acyclic graph (DAG) model to characterize the precedence constraints among jobs. Based on the proposed models, we jointly optimize the allocated VMs and the assigned workload to minimize the total resource cost under the response time constraints. Since the formulated problem is mixed integer non-linear programming, a heuristic is proposed to efficiently allocate resources for practical services. Experimental results show that the proposed scheme can effectively allocate VMs and schedule workload to achieve the minimal resource cost.

I. INTRODUCTION

Recent years have witnessed the rapid development of cloud computing. Compared with the conventional computing paradigms, an important feature of cloud computing is to enable the rapid and elastic resource provisioning. In cloud data centers, a shared pool of servers are managed to provide on-demand computation, communication, and storage resources as utilities in a scalable manner [1]. To efficiently provide resources, virtualization techniques have been applied to package computation resources into virtual machines (VMs). By managing VMs, cloud computing is able to provision or release resources in a fine granularity[2].

The emergence of cloud computing attracts multimedia service providers (MSPs) to deploy services on cloud. As one example, Netflix [3], one major Internet media streaming provider, moved its streaming services to Amazon Web Service (AWS) public cloud. From MSP’S perspective, the elastic and on-demand resource provision in cloud can effectively satisfy the intensive demands of multimedia services. By leasing VMs from cloud providers, MSPs can avoid the overhead cost on expensive servers.

There are two fundamental challenges faced by MSPs: one is resource allocation, and the other is workload scheduling. When deploying services, MSPs have to determine the required VM instances to support the thousands of user requests. But it is difficult to achieve the optimal resource allocation. First of all, multimedia services have strict response time requirements. A low response time means a fast response to user’s request, while a high response time will degrade user experience. If too many VM instances are allocated to one service, the service can be sped up, but the cost will be increased. On the other hand, if less instances are allocated, the service response time cannot be guaranteed. Thus, MSPs have to avoid both under-provisioning and over-provisioning.

In addition, cloud providers, like AWS, have a number of different VM classes. Different classes of VM instances have different resource capacities in terms of the computing units, CPU frequency, memory size, storage, and I/O rates. It is challenging for MSPs to determine the optimal amount and classes of instances.

Besides resource allocation, workload scheduling is another challenge. There are two levels of workload scheduling in cloud. The first level is request scheduling, in which users’ requests are distributed to different VMs to balance workload. The request scheduling can avoid service congestions. Compared to the request scheduling, the job scheduling is performed in a finer granularity. A multimedia service is generally composed of a set of jobs. Some jobs can run in parallel, while some jobs must be processed serially. The objective of job scheduling is to assign jobs on the allocated VMs to reduce the total execution time. Due to the varying workload and the precedence constraints among jobs, it is challenging to schedule workload to best utilize the allocated resources.

In this paper, we study cloud resource allocation and workload scheduling problem from MSPs’ perspective. The major contributions can be summarized as follows. We first introduce a queuing model to represent cloud services, and use a directed acyclic graph (DAG) model to characterize the precedence constraints among jobs. Based on these models, we jointly optimize the allocated VMs and the assigned workload to minimize the overall resource cost under the service response time constraints. Moreover, we propose a heuristic to efficiently allocate resources in practice. Experimental results demonstrate that the proposed resource allocation scheme can optimally allocate VMs and assign workload to achieve the
The emergence of cloud computing has introduced a significant revolution to the traditional IT industry. Today, the most popular video-on-demand services, like Netflix [3], Youtube [4], social network services, like Facebook [5], and on-demand gaming services, like Xbox on demand [6], are all deployed on cloud data centers.

Cloud resource allocation has always been a challenging research topic. Wu et al. [7] presented a cloud-based video on demand (VoD) system, and investigated the relationship between the dynamic viewing behavior and the allocated cloud resources. Wang et al. [8] proposed a cloud-based live media streaming framework, which adaptively leased or adjusted cloud resources according to the temporal or spatial dynamics of user demands. Miao et al. [9] studied the resource allocation for cloud-based FVV system. Niu et al. [10] explored the cloud bandwidth allocation and pricing strategies.

Besides resource allocation, workload scheduling is another challenging topic. Hui et al. [11] presented a cloud load balancing technique to allocate the required resources to different applications in the shortest time. Zhu et al. proposed the cloud-based PhotoSynth service [12], where they explored the parallelization on the request level and the task level, respectively. Yassa et al. [13] formulated the workload scheduling problem as a constraint optimization problem with multiple objectives, including the execution time, resource cost, and energy consumption. In our previous work [14], [15], we optimized cloud workload scheduling at the request level and the job level based on the given resources. However, the dynamic resource allocation has not been considered in [14], [15]. Compared to previous work, our study is different in the following senses: 1) we propose a queueing model to quantify the service performance and a DAG model to characterize the precedence constraints among jobs; 2) we jointly optimize cloud resource allocation and workload scheduling to minimize the overall resource cost under response time constraints.

III. SYSTEM MODELS

In this section, we present the proposed system models, including the resource allocation architecture, the queueing model, and the DAG model.

In cloud-based multimedia service, MSPs lease a collection of different classes of VMs. VMs can work together as the virtual cluster [16] to provide the more powerful resources. Fig. 1 presented the resource allocation architecture. The service process is described as follows. When requests are sent to cloud data center, the request scheduler will distribute user requests to different clusters for processing. The workload monitor performs a live monitoring on the type and number of requests, and forwards the information to the resource allocation module. With the request arrival rate, price rate, as well as response time requirements, the proposed resource allocation scheme will optimally allocate the required VMs to each cluster to achieve the minimal resource cost. Inside each cluster, the job scheduler will assign jobs to different VMs for distributed processing. If the initially reserved VMs cannot satisfy the resource demands, the resource allocation will request additional on-demand VMs from cloud providers to guarantee the service quality.

We use a queueing model to represent the cloud service procedure. The request arrival at a cloud service maps to the customer arrival at a queue, and the accomplishment of the requested service equals to the completion of the service in the queue. The allocated VMs are represented by the servers in each queueing system. It should be noted that the service rate must be higher than the request arrival rate to avoid that the queue eventually grows to infinity. In practice, the actual allocated cloud resources should satisfy the resource demands, otherwise the local congestion will occur in the service.

Fig. 2 presented the proposed queueing model. Suppose that there are $M$ virtual clusters to provide service. Let $S$ be the request scheduler, and $\mu_i^{(t)}$ be the mean service rate of cluster-$i$ at time $t$. According to [17], the arrivals of requests can be modeled as a Poisson Process with the mean arrival rate of $\lambda(t)$. As shown in Fig. 2, the incoming requests are scheduled to the cluster-$i$ with weight $\omega_i^{(t)}$. According to the decomposition property of Poisson distribution [18], the arrivals of the scheduled requests to cluster-$i$ also follow a Poisson Process with the mean arrival rate of $\omega_i^{(t)}\lambda(t)$. Thus, the service process at cluster-$i$ can be modeled as an $M/M/1$ queueing system [18]. To ensure the queueing system stable, $\omega_i^{(t)}\lambda(t) < \mu_i^{(t)}$ is required.

A multimedia service consists of a number of jobs with
We introduce a directed acyclic graph (DAG) model to characterize the precedence constraints. A DAG is a directed graph with no path that starts and ends at the same vertex. The DAG model is denoted as: \( \text{DAG} = (V, E) \), where \( V \) is the set of vertices and \( E \) is the set of edges. Each vertex represents a job, while each edge characterizes a precedence constraint between two jobs. For example, the edge \( e_{\Phi_k\Phi_k} \) represents that job \( \Phi_k \) cannot be executed until job \( \Phi_k' \) has finished. Fig. 3 illustrates a 3D scene reconstruction framework and the corresponding DAG model. In the framework, there are 5 types of jobs, and the request and the result are denoted as the source and the sink node. From Fig. 3, we can find that there are multiple paths from source to sink. The service execution time is determined by the longest path in the DAG. The target of job scheduling is to optimally schedule jobs to VMs, so that the total execution time can be minimized.

IV. JOINT OPTIMIZATION OF RESOURCE ALLOCATION AND WORKLOAD SCHEDULING

In this section, we investigate the resource allocation and workload scheduling problem for cloud based multimedia services. Our objective is to jointly optimize the allocated VM instances and the assigned workload to achieve the minimal resource cost.

Suppose that \( N \) classes of VM instances with different configurations are supplied by the cloud provider. Currently, there are two major VM pricing plans, i.e. the reservation plan and the on-demand plan [19]. In the reservation plan, MSPs reserve a VM instance for a long term and in turn receive a discount on the hourly rate. In the on-demand plan, the VMs can be paid by the hourly usage without long-term commitment. The price rate in the on-demand plan is much higher than that in the reservation plan. Let \( P^e_j \) and \( P^d_j \) be the price rates of class-\( j \) VM instance in the reservation plan and on-demand plan, respectively. Let \( X^{e(t)}_{ij} \) and \( X^{d(t)}_{ij} \) denote the number of reserved and on-demand class-\( j \) VMs allocated to cluster-\( i \) at time \( t \). Thus, the total resource cost can be formulated as

\[
C^{(t)} = \sum_{i=1}^{M} \sum_{j=1}^{N} P^e_j X^{e(t)}_{ij} + \sum_{i=1}^{M} \sum_{j=1}^{N} P^d_j X^{d(t)}_{ij},
\]

where \( \sum_{i=1}^{M} \sum_{j=1}^{N} P^e_j X^{e(t)}_{ij} \) is the cost of reserved VMs, and \( \sum_{i=1}^{M} \sum_{j=1}^{N} P^d_j X^{d(t)}_{ij} \) is the cost of on-demand VMs.

We take the response time as the QoS metric. Based on the analysis in Sec. III, the service at cluster-\( i \) can be modeled as a \( M/M/1 \) queueing system. Thus, the response time \( T^{(t)}_i \) at cluster-\( i \) is given by [18] \( T^{(t)}_i = \frac{1}{\mu^{(t)}_i - \omega^{(t)}_i \lambda^{(t)}}. \) The mean response time in cloud is therefore formulated as

\[
T^{(t)} = \sum_{i=1}^{M} \omega^{(t)}_i T^{(t)}_i = \sum_{i=1}^{M} \frac{\omega^{(t)}_i}{\mu^{(t)}_i - \omega^{(t)}_i \lambda^{(t)}}, \quad (2)
\]

where \( \mu^{(t)}_i \) is the average service rate at cluster-\( i \).

To effectively utilize the allocated resources, the job scheduler in each cluster will assign jobs to VMs for the distributed processing. Suppose that the multimedia service can be decomposed into \( K \) jobs, which are denoted as \( \{\Phi_1, \Phi_2, \ldots, \Phi_K\} \). Some jobs can be executed concurrently, while some jobs have to be processed sequentially. Let \( X^{e(t)}_{ijk} \) and \( X^{d(t)}_{ijk} \) be the number of reserved and on-demand class-\( j \) instances allocated to job \( \Phi_k \) at cluster-\( i \), and \( v^e_j \) be the average service rate of using one class-\( j \) VM instance to process job \( \Phi_k \). Thus, the processing time of job \( \Phi_k \) at cluster-\( i \) is formulated as

\[
t^{(t)}_{ik} = \frac{1}{\sum_{j=1}^{N} \left( X^{e(t)}_{ijk} + X^{d(t)}_{ijk} \right)} v^e_j. \quad (3)
\]

According to Amdahl’s law [20], the speed-up of a program from parallelization is limited by the sequential portion of the program. In the DAG model, there are multiple paths from source to sink. The jobs in the same path must be executed sequentially, while different paths can be processed in a parallel way. So, the service execution time in cluster-\( i \) is determined by the time of the longest path from source to sink. Suppose that there are \( W \) paths in the DAG, and let \( \Phi_w \) denote the set of vertices on the path-\( w \) (\( \forall w = 1, \ldots, W \)). Thus, the execution time at cluster-\( i \) can be formulated as \( \sum_{k \in \Phi_w} T^{(t)}_{ik} \), and the total execution time at cluster-\( i \) is given by

\[
T^{(t)}_i = \max_{\{w \in W\}} \left\{ \sum_{k \in \Phi_w} T^{(t)}_{ik} \right\}, \quad (4)
\]

which represents the longest execution time among all paths. If more VM instances are allocated to cluster-\( i \), it will lead to a lower execution time. In queueing theory, the service rate is the reciprocal of the service time. Thus, the average service rate \( \mu^{(t)}_i \) at cluster-\( i \) is given by

\[
\mu^{(t)}_i = 1/T^{(t)}_i.
\]

\[
= 1/\max_{\{w \in W\}} \left\{ \sum_{k \in \Phi_w} T^{(t)}_{ik} \right\}, \quad (5)
\]

\[
= 1/\max_{\{w \in W\}} \left\{ \sum_{k \in \Phi_w} \sum_{j=1}^{N} \frac{1}{X^{e(t)}_{ijk} + X^{d(t)}_{ijk}} v^e_j \right\}.
\]
Based on the above analysis, the mean response time in Eq. (2) can therefore be formulated as

\[
T(t) = \sum_{i=1}^{M} \frac{\omega_i(t)}{1} - \omega_i(t)\lambda(t). \quad (6)
\]

Moreover, the total number of allocated class-\(j\) VMs in the reservation plan should be no more than the number of initially reserved class-\(j\) VMs. Let \(X_{j}^{ini}\) be the number of initially reserved class-\(j\) VMs. Thus, this constraint is given by

\[
\sum_{i=1}^{M} X_{ij}^{r(t)} \leq X_{j}^{ini}, \quad (\forall j = 1, 2, \ldots, N).
\]

With the above derivations, we can formulate the joint resource allocation and workload scheduling problem as follows.

\[
\text{Minimize} \quad C(t)
\]

\[
\text{subject to} \quad \begin{aligned}
T(t) & \leq \tau, \\
\omega_i(t)\lambda(t) & < \mu_i(t), \\
\sum_{k=1}^{K} X_{ijk}^{r(t)} & \leq X_{ij}^{r(t)}, \\
\sum_{k=1}^{K} X_{ijk}^{d(t)} & \leq X_{ij}^{d(t)}, \\
\sum_{i=1}^{M} X_{j}^{r(t)} & \leq X_{j}^{ini}, \\
\sum_{i=1}^{M} \omega_i(t) & = 1, \\
X_{ijk}, X_{ijk}^{d(t)} & \in \mathbb{N}, \omega_i(t) \in [0, 1],
\end{aligned} \quad (7)
\]

where \(C(t)\), given by Eq. (1), is the total resource cost, \(T(t)\), given by Eq. (6), is the mean response time in cloud, \(\mu_i(t)\), given by Eq. (5), is the average service rate at cluster-\(i\), and \(\tau\) is the upper bound of response time. The objective function in Eq. (7) is to minimize the overall resource cost. The constraint \(\tau \leq T(t)\) represents the service response time requirement, the constraints \(\omega_i(t)\lambda(t) < \mu_i(t)\) are the queueing stability constraints at cluster-\(i\), the third, fourth, and fifth constraints are the constraints of allocated VM instances, and the constraint \(\sum_{i=1}^{M} \omega_i(t) = 1\) is the workload conservation constraint, i.e., all workloads should be scheduled for processing. The optimization variables \(X_{ijk}^{r(t)}\) and \(X_{ijk}^{d(t)}\) are natural numbers, while the scheduling weights \(\omega_i(t)\) are real numbers in the range of \([0, 1]\).

The optimization problem in Eq. (7) is a mixed integer non-linear programming, which is known to be NP-hard [21]. The problem can be solved by the extensive search method [22]. However, the extensive search takes the exponential time complexity, which cannot quickly adapt to the dynamic workload. Thus, we propose a heuristic to efficiently allocate cloud resources and schedule workload for practical services. The proposed heuristic is based on the following three core ideas.

- We introduce the unit cost [23] as the metric to evaluate the economic performance in each class of instance. Specifically, the unit cost \(q_{jk}\) is defined as the cost of using one class-\(j\) VM instance to process one unit request of job-\(k\). Intuitively, if a VM instance has a lower unit cost, it means that the instance is more desirable and should be allocated to process the job first. Therefore, we precompute the unit cost of each class VM for each job, and rank them in ascending order. When allocating VM instances, we first allocate instances with the lowest unit cost.
- We apply the critical path method [24] to find the bottleneck in the service. Critical path is defined as the longest path in a DAG, and it determines the total time for completing the service. By speeding up the jobs on the critical path, we can effectively reduce the total execution time. According to [24], we can use dynamic programming to find the critical path in a DAG with the time complexity of \(O(|V| + |E|)\), where \(V\) is the number of vertices and \(E\) is the number edges in the DAG.
- We employ the utilization based greedy scheduling [18] to schedule user requests to different clusters. The utilization at cluster-\(i\) is given by \(\rho_i = \frac{\omega_i(t)\lambda(t)}{\mu_i(t)}\), which represents the fraction of time that the cluster-\(i\) is busy. In the heuristic, we compute the utilization of each cluster and schedule user request to the cluster with the lowest utilization. To efficiently maintain the dynamic resource utilization, we apply a min heap [25] to find the lowest utilization with \(O(\log M)\) complexity, where \(M\) is the number of clusters.

Based on the above discussions, we presented the proposed heuristic for resource allocation and workload scheduling in Algorithm 1. The proposed heuristic firstly schedules user requests among clusters with the utilization based greedy scheduling, secondly finds the critical path using the dynamic programming, and finally allocates resources according to unit cost to satisfy the response time requirement. Moreover, the proposed heuristic can be performed parallelly at each cluster after the request scheduling.

V. PERFORMANCE EVALUATION

In this section, we evaluate the proposed resource allocation and workload scheduling scheme. The evaluation includes the numerical simulation and prototype test.

A. Numerical Simulation

Amazon Web Service (AWS) [19] is one of the most popular cloud computing platforms, allowing MSPs to lease VMs for various services. A large number of service providers, like Netflix, Airbnb, Expedia, and Adobe, choose AWS to deploy their services. To make our simulations convincing, we apply the price rates and VM configurations of AWS in our simulations. We tested four classes of VM instances, including general purpose instances (t2 mediums and t2. large), computation optimized instances (c3. xlarge), and GPU instances (g2.2xlarge). The detailed configuration and price rates can be found from [26]. In the simulation, we apply the 3D scene reconstruction as the cloud-based multimedia service. Users can submit a video clip to cloud server, and receive the reconstructed 3D point cloud. We set three clusters to provide service and the response time upper bound \(\tau = 3\) seconds.
Currently, the state-of-the-art resource allocation scheme is the utilization scheme, which has been applied by Amazon Elastic Beanstalk [27]. The utilization scheme adjusts cloud resources based on the resource utilization threshold, where the VM provisioning will be triggered when utilization \( \rho \) is higher than \( \delta_0 \) and stopped until utilization \( \rho \) is lower or equal to \( \delta_1 \) (\( \delta_1 < \delta_0 \)). In the simulation, we compare the resource cost among the following alternative schemes: 1) the optimal allocation scheme, where the allocated VMs and scheduled workload are acquired by solving the optimization problem in Eq. (7) with extensive search, 2) the proposed heuristic in Algorithm 1, 3) and the utilization scheme [27] with \( \delta_1 = 0.5 \) and \( \delta_0 = 0.7 \). Fig. 4 presents the comparison of resource cost among the three schemes when \( \lambda \) varies from 100 requests/s to 1000 requests/s. From Fig. 4, we can see that the optimal allocation scheme can achieve the lowest resource cost, and the proposed heuristic can provide satisfactory services at a lower resource cost compared with the utilization scheme. It should be noted that the optimal allocation scheme is achieved by extensively search the whole feasible region, which is globally optimal benchmark but not efficient, while the proposed heuristic is near optimal but lightweight and practical. Moreover, we can find that the total resource cost acquired by the proposed heuristic is close to the globally optimal solution. Thus, the proposed heuristic can approach the optimal allocation scheme in an efficient way. The utilization scheme [27] only guarantees that the utilization at each cluster is lower than the upper bound. However, it fails to assign the most suitable VMs to process each job, and thus leads to a high resource cost. When the request arrival rate \( \lambda = 1000 \) requests/s, the proposed heuristic saves 32% resource cost, compared with the utilization scheme.

Next, we evaluate the workload scheduling performance of the proposed scheme. In the simulation, we allocate the same amount and classes of VM instances to the three schemes, and compare the service response time under the varying workload. The comparison results of the average response time among the three schemes are shown in Fig. 5. From Fig. 5, we can find that the optimal allocation scheme achieves the lowest response time among the three schemes. The optimal allocation scheme is the globally optimal benchmark, while the proposed heuristic is a sub-optimal but lightweight solution. We can see that the proposed heuristic achieves the lower response time compared to the utilization scheme under the same request arrival rate. Fig. 5 also shows that the proposed heuristic performs close to the optimal allocation scheme. When the request arrival rate is 1000 requests/s, the difference of the response time between the two schemes is 0.032 seconds.

B. Prototype Test

Besides numerical simulations, we also develop a prototype to test the proposed resource allocation and workload scheduling scheme. We use a server with Intel i7 CPU 3.07GHz, 16G RAM, and NVIDIA GeForce GTX 660, as the cloud server, and set up OpenStack [28] as the cloud platform to manage and provide VM instances. A VM instance is employed to run the proposed heuristic to allocated resource and schedule workload. We deploy the 3D scene reconstruction framework on the cloud, and generate workload to request for service.

Since the optimal allocation scheme performs extensive search to acquire the optimal solution, it cannot adapt to the varying workload in practice. Thus, we compare the performance between the proposed heuristic and the utilization

---

Algorithm 1: Heuristic for Resource Allocation and Workload Scheduling

1. **Preprocess:** Compute unit costs \( q_{jk}^r = \frac{\mu_j^r}{\lambda_j} \) and \( q_{jk}^d = \frac{\mu_j^d}{\lambda_j} \), which are the unit costs of using one reserved and one on-demand class-\( j \) VM instance to process one job-\( k \) request, respectively. Let set \( Q = \{ q_{jk}^r, q_{jk}^d \mid j = 1, \ldots, N, k = 1, \ldots, K \} \) and sort set \( Q \) in ascending order.
2. Compute the resource utilization \( \rho_i^t = \frac{\omega_i^t}{\omega_i^d}, \) (\( \forall i = 1, \ldots, M \)), and insert \( \rho_i^t \) into a min heap \( H \) [25]. Calculate the average utilization \( \rho^t = \sum_{i=1}^{M} \rho_i^t / M \).
3. **while** there are user requests to be scheduled **do**
   4. Get the cluster-\( i \) with the minimal utilization from the root of the min heap \( H \).
   5. Schedule requests to the cluster-\( i \) until \( \rho_i^t = \rho^t \).
   6. **end while**
   7. **for** each cluster-\( i \) (\( \forall i = 1, \ldots, M \)) **do**
      8. **repeat**
         9. Find the critical path \( \Phi_i \) in the DAG using dynamic programming [24].
         10. Compute the total execution time \( T_i^c(t) \).
         11. Allocate VMs with the lowest unit cost from set \( Q \) to each job \( V_k \) on the critical path.
         12. **if** the allocated class-\( j \) VMs are from the reserved VMs, then the allocated amount is no more than \( X_j^{ini} \).
            13. **until** the response time \( \frac{\omega_i^t}{\rho_i^t} - \frac{\omega_i^d}{\lambda_j} \leq \tau \).
            14. **end for**
      15. **return** the total resource cost \( C(t) \).

---

Fig. 4. Comparison of resource cost when \( \lambda \) varies from 100 requests/s to 1000 requests/s.
scheme in the test. Fig. 6(a) presents the comparison results of resource cost. Compared with the utilization scheme, the proposed heuristic can serve user requests at a lower resource cost. We analyze the reason behind the difference. The utilization scheme demands additional VM instances, once the resource utilization is higher than the threshold $\delta_t$. Thus, there exists resource waste in the utilization scheme. In the proposed heuristic, the request scheduler dynamically distributes user requests according to the varying resource utilization. So, the proposed heuristic can fully utilize the allocated resources without service congestion. To give a clear view, we compare the average resource utilization in Fig. 6(b). As shown in Fig. 6(b), the proposed heuristic achieves a higher resource utilization than the utilization scheme.

VI. CONCLUSIONS

In this paper, we study the resource allocation and workload scheduling problem for cloud-based multimedia services. Specifically, we introduce a queuing model to represent the service procedure, and a directed acyclic graph (DAG) model to characterize the precedence constraints among jobs. Based on the proposed models, we jointly optimize the allocated VMs and the assigned workload to minimize the total resource cost under the response time constraints. Since the formulated optimization problem is a mixed integer non-linear programming, we propose a heuristic to efficiently allocate resources and schedule workload for practical services. Experimental results demonstrate that the proposed scheme can effectively allocate VMs and schedule workload to achieve the minimal resource cost.

ACKNOWLEDGMENT

This work is supported by NSERC grant CRDPJ 436798-12.

REFERENCES