Resource Provisioning and Profit Maximization for Transcoding in Clouds: A Two-Timescale Approach

Guanyu Gao, Han Hu, Yonggang Wen, *Senior Member, IEEE,* Cedric Westphal, *Senior Member, IEEE*

***Abstract*—Transcoding is widely adopted for content adaptation, however, it may incur excessive resource consumption and processing delays. Taking advantage of cloud infrastructure, cloud-based transcoding can elastically allocate resources under time-varying workloads and perform multiple transcodings in parallel to reduce delays. To provide transcoding as a cloud service, cloud transcoding systems require some intelligent mechanisms to provision resources and schedule tasks to satisfy user requirements while maximizing financial profit. To this end, we propose a two-timescale stochastic optimization framework for maximizing service profit while achieving performance requirements by jointly provisioning resources and scheduling tasks under a hierarchical control architecture. Our method analytically integrates service revenue, processing delay, and resource consumption in one optimization framework. We derive the offline exact solution, and design some approximate online solutions for task scheduling and resource provisioning. We implement an open source cloud transcoding system, called *Morph*, and evaluate the performance of our method in a real environment. Empirical studies verify that our method can reduce resource consumption and achieve a higher profit compared with baseline schemes.**

***Index Terms*—Transcoding, cloud computing, resource provisioning, scheduling, profit maximization**

I. Introduction

Transcoding is widely adopted for content adaptation [1, 2]. Videos are commonly transcoded into multiple representations in different bitrates for streaming to adapt to diverse user devices and varying network conditions [3–5]. However, transcoding is very compute intensive [6]. Content producers need to maintain many in-house servers to transcode large volumes of videos. Moreover, in-house solutions typically require over-provisioning computing resources by at least 30% to meet peak workloads [7]. This wastes massive resources in normal workloads. Transcoding is also very time consuming. It may takes several hours to transcode a large video such as a movie or a TV show. This may incur intolerable delays for videos that must be delivered timely.

Cloud computing introduces a new way for transcoding, and content producers are switching to cloud-based transcoding as a new solution. Leveraging cloud infrastructure, cloud transcoding systems can elastically provision and release resources commensurate with workloads, and this can avoid resource wastage [8]. Meanwhile, with a large number of available virtual machines (VMs), it can perform multiple transcodings in parallel for a video using many VMs, and this can greatly reduce transcoding delays. Cloud-based transcoding can cut costs for content producers and help them focus on creating content instead of being impeded by IT obstacles.

To provide transcoding as a cloud service, there still exist some challenges to be addressed. *First*, as transcoding workloads are time-varying, if resources are over-provisioned, it wastes resources; and if resources are under-provisioned, it incurs long delays and deteriorates service quality. Cloud transcoding systems must intelligently provision right amount of resources under time-varying workloads. *Second*, users have different performance requirements, and cloud transcoding systems must schedule tasks strategically to meet performance requirements. *Third*, cloud transcoding service providers are keen on financial profit. The resource provisioning policy and task scheduling policy in a cloud transcoding system are key players in system performance and cost, which greatly affects service profit. Therefore, one must take service profit into account when designing these policies to reduce financial risk.

Many works have considered the problems of resource provisioning and task scheduling in the data centre or cloud. [9–11] considered the strategies for provisioning resources under dynamic workloads. These proposed methods can dynamically provision right amount of resources by managing the tradeoff between resource consumption and delays. However, these works do not consider different performance requirements of tasks, and thus, the proposed methods only schedule tasks with the first-in, first-out (FIFO) discipline. Meanwhile, these works do not consider divisible tasks, which can be divided into multiple sub-tasks and processed on multiple VMs in parallel. However, this problem is important for transcoding, because the transcoding time for a video can be extremely long if the video is processed only by one VM. [12–15] considered profit maximization problems, yet these works do not consider the task scheduling problem for maximizing profit.

We aim to maximize profit for the service provider while meeting users' performance requirements. Service profit is an important consideration when delivering transcoding as an online service, because the service provider must reduce the financial risk of operating a service. To make the service profitable, not only the processing delay and resource consumption should be considered, but also service revenue. We integrate service revenue, resource consumption, and processing delay in a two-timescale stochastic optimization framework. The system is controlled under a hierarchical control architecture, where resource provisioning actions are performed at a lower

G.Y. Gao is with the Interdisciplinary Graduate School, Nanyang Technological University, Singapore. H. Hu and Y.G. Wen are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. C. Westphal is with Huawei Innovation Center, USA, and Department of Computer Engineering, University of California, Santa Cruz, CA. E-mail: {ggao001, hhu, ygwen}@ntu.edu.sg, cwestphal@huawei.com, cedric@soe.ucsc.edu.

frequency to accommodate time-varying workloads, while task scheduling actions are performed at a higher frequency to maximize service profit and meet tasks' performance requirements. We derive the offline optimal solution, and design some online approximate solutions for system control. To evaluate the performance of our method, we design and implement an open source cloud transcoding system, *Morph*, following the master-worker paradigm. Experiment results demonstrate that our method can reduce resource consumption while achieving a higher profit compared with baseline schemes. Although our work is specific to cloud-based transcoding, our proposed framework would be also applicable to other cloud services.

Our contributions in this paper are summarized as follows:

- A two-timescale stochastic optimization framework for maximizing service profit while achieving performance requirements by jointly scheduling tasks and provisioning resources under a hierarchical control architecture.
- A neural network method for estimating the required computing resources of a transcoding task.
- The design and implementation of an open source cloud transcoding system, *Morph*[1], and the performance evaluation of our method in a real environment.

The rest of this paper is organized as follows. Section II introduces related works. Section III presents the system design. Section IV describes the system models and problem formulation. Section V derives the exact offline solution. Section VI designs some approximate online solutions. Section VII demonstrates the system implementation and performance evaluation. Section VIII concludes the paper.

II. Related Work

In this section, we present related works and discuss the differences of our method compared with existing works.

Resource provisioning. Many works have studied transcoding systems. In our previous work [16, 17], we presented the design of our cloud transcoding system, Morph, and its resource management mechanism. [18] proposed a method to jointly perform video transcoding and delivery in an online manner. [9, 19] studied the strategies for provisioning right amount of resources under time-varying workloads. Both works only considered tasks with homogenous performance requirements, and did not consider the scheduling problem for tasks with different performance requirements. Meanwhile, these works did not consider the scheduling problems for the video blocks of a task on multiple VMs in parallel. [20–22] studied the problem of scheduling tasks under a fixed amount of resources. However, the workloads in online services are time-varying, it cannot achieve resource efficiency and desired performance requirements without scaling computing capacities to accommodate time-varying workloads. [23–25] proposed transcoding mechanisms to reduce streaming delays, and these works did not consider resource provisioning problems.

Our work mainly focuses on the problems of task scheduling and resource provisioning specific to cloud-based transcoding, while some other works have studied these problems in the data center [10, 11] or cloud [26–29]. These works studied the dynamic resource provisioning problem under time-varying workloads. [10, 11, 28] studied the resource provisioning problem for achieving the prescribed QoS criteria or reducing the overall cost, yet the tasks considered in these works are homogenous. [26, 27] studied the heterogeneity-aware resource provisioning problem, but they did not consider the problem of scheduling the pending tasks with different priority levels. Compared with these works, we also study the problem of scheduling a task on multiple VMs in parallel to reduce processing delays.

Profit maximization. The service profit maximization problem has been intensively studied in the data center [12, 15, 30], cloud computing [13, 14], and wireless networks [31–33]. [12] considered the profit maximization problem of scheduling a deterministic set of tasks. [31] studied how to assign resources to a number of clients so that the total utility can be maximized. The problems studied in [12, 31] are deterministic, while this does not suit the stochastic task arrivals in cloud. [14] studied the resource provisioning problem for each user to maximize per-user financial profit. [15, 32] studied the dynamic pricing problem in data center and wireless networks to maximize profit. [30] studied the profit maximization problem in data center by considering the fluctuations of electricity price. [14, 15, 30, 32] achieved the profit maximization from different perspectives according to their problem scenarios. [13] studied the profit maximization problem in cloud by modeling the system as the M/M/m queueing model. The differences of our work is that we consider the profit maximizing problem for scheduling tasks with different performance requirements and scheduling a task on multiple VMs in parallel.

Our method manages the service based on deadline-dependent pricing for maximizing profit. Minimizing resource consumption while guaranteeing average delays is also widely adopted for performance management. However, the average delay is only statistically guaranteed among users. The processing delay for a specific task may scatter in a wide range of values. It will frustrate a user if processing delays are long, but charged price is the same. An advantage of our method is that users are charged less with longer delays, and this impels the service provider to provide a better service to improve revenue. Users will also be charged a higher price for a low processing delay or a high priority level service. Therefore, it can balance the service provider's desire for service profit and users' performance requirements.

III. System Design

In this section, we introduce the design of our cloud transcoding system and the workflows.

A. Architecture

We illustrate the system architecture in Fig. 1. The system consists of the following modules.

Service Interface: It estimates the required computing resource for a task and segments video files into blocks according to the group of pictures (GOP) structure.

[1] https://github.com/cap-ntu/Morph.

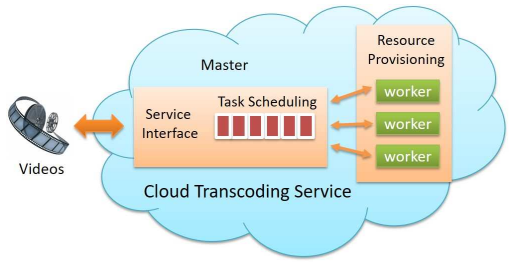


Fig. 1. The system architecture of our cloud transcoding system.

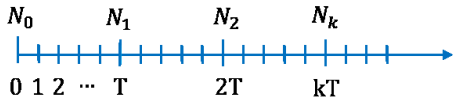


Fig. 2. The discrete time model. The time horizon is divided into two timescales. The fast timescale is denoted as $t = 0, 1, 2, ...$, and the slow timescale is denoted as $N_0, N_1, N_2, ...$

Resource Provisioning: It manages many VMs[2], and each VM runs a transcoding worker. Whenever a worker is idle, it will fetch a video block from the master and transcode the video block into the target representation. The resource provisioning module dynamically controls the number of active VMs according to system workloads.

Task Scheduling: It maintains a queue for pending tasks and sequences their order according to the task scheduling policy. When there is a request for a video block from a worker, the scheduler will pick a video block of the head-of-queue task to dispatch. The video blocks of a task are scheduled as a chain. Once the scheduler decides to process a video block of a task (i.e., a chain), it has to complete all of the video blocks of the task (i.e., the entire chain) before working on the next task.

B. Workflows

The system workflows are as follows. *First,* content producers upload videos and specify transcoding requirements. The system estimates the required computing resource for each task and segment videos into blocks, ensuring that each block consumes the same computing time. *Second,* the task scheduler sequences pending tasks according to the task scheduling policy. *Third,* when a worker is idle, it will request a video block from the master. After transcoding a video block into the target representation, the worker will send back the video block of the target representation to the master. The master will concentrate the video blocks of a video into one file.

IV. System Model

In this section, we introduce the system models and problem formulation. The key notations are summarized in Table I.

A. Task Arrival Model

We adopt a discrete time model and divide the time horizon into two timescales. As illustrated in Fig. 2, we denote the time in the fast timescale as $t = 0, 1, 2, ...$, and the time in the slow timescale as N_k, where $k = 0, 1, 2, ...$ N_k is from time slot kT to time slot $(k+1)T - 1$, which consists of T time slots. The typical length of one time slot is $1 \sim 10$ seconds and T is $1800 \sim 3600$ seconds. We model the task arrival as a non-stationary process with different arrival rates over the slow timescale. For each time slot within N_k, we assume the task arrival distribution is homogenous. We denote the task arrival rate during a time slot within N_k as λ_k.

[2]It can also adopt the Container technology for resource virtualization.

TABLE I
Key Notations

Notation	Definition
t	Time in the fast timescale, $t = 0, 1, 2, ...$
N_k	Time in the slow timescale, $k = 0, 1, 2, ...$
λ_k	Task arrival rate in each time slot of N_k.
D_i	Estimated transcoding time for task i.
$U_i(t)$	Pricing function of task i.
$P(t)$	Service revenue at time slot t.
$C^v(N_k)$	Cost for renting VMs during N_k.
ν_k	Resource provisioning action at N_k.
ψ_k^s	State in the slow timescale at beginning of N_k.
ψ_t^f	State in the fast timescale at time slot t.
R_k^s	Service revenue over T time slots of N_k.
π^s	Resource provisioning policy in the slow timescale.
π^f	Task scheduling policy in the fast timescale.
f_i	Estimated completion time of task i.
$V(\psi^s, \psi^f)$	Expected overall profit starting from state (ψ^s, ψ^f).
$Q(\phi, \nu)$	Action-value function for Q learning algorithm.

B. Required Computing Resource Estimation Model

We estimate the required computing resource (i.e., consumed computing time) of each task for task scheduling. Transcoding is a complicated conversion of one coded signal to another [34], and the consumed computing time depends on many factors, e.g., video resolution, bitrate, and duration, etc. We adopt the neural network method for learning the non-linear relation between the transcoding time of a task and other dependent factors. We adopt a three-layer feedforward neural network, which consists of the input layer, hidden layer, and output layer. We use the original video duration, bitrate, frame rate, resolution, and the target resolution as the input feature vector. The output layer generates the estimated transcoding time of a task. The neural network is trained offline. We use the feature vector of a task as the input, and the neural network generates the estimated transcoding time of the task.

C. Service Revenue Model

The service provider can negotiate with users to determine the charged price based on the performance and consumed resources. This service contract between users and the service provider can be formally established as a service level agreement (SLA). With a SLA, a balance point between the service provider and users can be determined to maximize profit while keeping users satisfied. We adopt a deadline-dependent pricing model which charges users based on consumed computing resource, task completion time, and service priority level. If a task is completed with a longer delay, the user will be charged less, and this can impel the service provider to provide a better

quality service to improve its revenue. Users will be charged a higher price for a low processing delay or a high priority level service. This can balance the service provider's desire for profit and users' performance requirements.

One possible pricing function is given in Eq. 1. Suppose task i is submitted at time a_i and completed at t. The charged price for the task will be

$$U_i(t) = \alpha^{t-a_i} R_i D_i, 0 < \alpha < 1, t \geq a_i, \tag{1}$$

where α is the discounting factor, R_i is the marginal price for one unit of consumed computing time, and D_i is the overall consumed computing time. If a task consumes more computing resource, it will be charged a higher price. The marginal price is determined by the priority level of a task. A high priority corresponds to a higher marginal price. If a task is given a higher priority, it will be charged a higher price. The discounting factor reflects the deadline-dependent pricing. If a task is completed faster, the charged price will be higher and the service provider can gain a higher revenue.

The pricing function affects the task scheduling policy. Our framework are also applicable to some other functions, e.g., linear function and step function. The linear function is

$$U'_i(t) = w_i - \beta_i(t - a_i), t \geq a_i, \beta_i > 0, \tag{1.a}$$

where w_i is the initial price for task i and β_i is the discounting factor. When $U'_i(t)$ is less than zero, it can be seen as penalty for excessive delays. The step pricing function is

$$U''_i(t) = \begin{cases} w_i, & a_i \leq t \leq a_i + \tau_i, \\ 0, & t \geq a_i + \tau_i, \end{cases} \tag{1.b}$$

where τ_i is the prescribed deadline for task i. If a task cannot meet the deadline, it will not be charged.

Let the completion time of task i be f_i and the set of tasks at time slot t be x_t, the service revenue at t is

$$P(t) = \sum_i U_i(t), \{i \mid f_i = t \cap i \in x_t\}. \tag{2}$$

Because available resources are limited and revenue decreases with processing delays, the task scheduler must schedule pending tasks strategically to maximize overall revenue.

D. Computing Cost Model

The computing cost is incurred by provisioning VMs. If provisioned resources exceed real demands, it will incur unnecessary cost and reduce profit. On the contrary, if provisioned resources are less than required, it cannot satisfy the performance requirement and will incur long delays, which will also reduces revenue. We scale the system each T time slots to meet the time-varying workloads. We assume that the VMs are homogeneous and the overall cost is proportional to the number of VMs. Let the number of provisioned VMs during N_k be $M(N_k)$. The computing cost during N_k is

$$C^v(N_k) = M(N_k) C_v, \tag{3}$$

where C_v is the cost for one VM over T time slots.

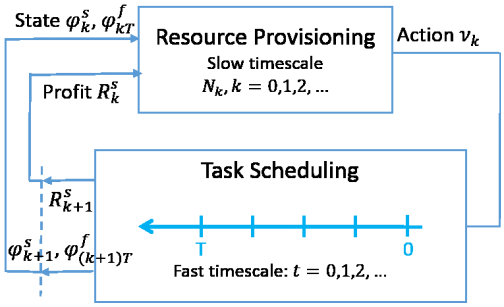


Fig. 3. The two-timescale optimization framework for profit maximization.

E. Service Profit Maximization

Our objective is to maximize profit over a long run. The system dynamics can be characterized in two timescales [35–37]. The system provisions resources at a lower frequency in the slow timescale while scheduling incoming tasks at a higher frequency in the fast timescale. Specifically, at each time slot t, the system schedules pending tasks; at the beginning of each N_k, the system scales computing capacity.

We illustrate the system dynamics in Fig. 3. We define the system state in the slow timescale at the beginning of each N_k as $\psi_k^s = \{\lambda_k, m_{k-1}\}$, where λ_k is the task arrival rate during each time slot of N_k and m_{k-1} is the number of active VMs during N_{k-1}. Let the system state space in the slow timescale be Ψ^s. For each time slot t within N_k, we define the system state in the fast timescale as $\psi_t^f = \{x_t\}$, where x_t is the set of pending tasks at time slot t. For each task, we have its submission time and required computing resource. We let the state space in the fast timescale be Ψ^f.

The system determines the number of active VMs at the beginning of each N_k according to the system states in the slow and fast timescales. We denote the resource provisioning policy in the slow timescale as π^s and the resource provisioning action at N_k as ν_k. Specifically, $\nu_k > 0$ represents the number of new activated VMs and $\nu_k < 0$ represents the number of shutdown VMs. The action space for the resource provisioning action is denoted as $\wedge$. The mapping from the system states in the slow and fast timescales to the resource provisioning action by applying the policy π^s is

$$\pi^s : (\psi_k^s, \psi_{kT}^f) \rightarrow \nu_k, k = 0, 1, 2, \ldots. \tag{4}$$

The number of active VMs after taking the resource provisioning action is m_k, where $m_k = m_{k-1} + \nu_k \geq 0$.

The task scheduling policy in the fast timescale determines the transcoding order of pending tasks to maximize overall revenue. The system state in the fast timescale evolves over T time slots until the system state in the slow timescale changes. The system dynamics in the fast timescale is an MDP over finite T-horizon and the task scheduling policy is a sequence of T-horizon nonstationary policies. We define the sequence of task scheduling policies over the finite T-horizon as

$$\pi_T^f = \{\pi_T^f(kT), \pi_T^f(kT+1), ..., \pi_T^f((k+1)T-1)\}, \tag{5}$$

where $\pi_T^f(t)$ is the task scheduling policy at time slot t. We assume that the set of task scheduling policies is finite and the

same for each k. At each time slot t, we define the mapping from the system states in the two timescales and the action in the slow timescale to the task scheduling operation as

$$\pi_T^f(t) : (\psi_k^s, \nu_k, \psi_t^f) \rightarrow \ell_t,\ kT \leq t < (k+1)T, \quad (6)$$

where ℓ_t is the task scheduling action, i.e., the scheduled transcoding order of pending tasks.

Given slow timescale state ψ_k^s, resource provisioning action ν_k, fast timescale state ψ_{kT}^f, and T-horizon task scheduling policy π_T^f, the expected profit over T time slots during N_k can be calculated as

$$R_k^s(\psi_k^s, \psi_{kT}^f, \nu_k, \pi_T^f) = \mathbb{E}_{\psi_t^f}\left\{ \sum_{t=kT}^{(k+1)T-1} P(t) - C^v(N_k) \right\}.$$

The profit is the revenue over T time slots while deducting the computing cost. We aim to maximize profit by deriving task scheduling policy and resource provisioning policy. We present the service profit maximization problem as

$$\mathcal{P}1: \max_{\pi^s \in \Pi^s} \max_{\pi_T^f \in \Pi_T^f} \mathbb{E}_{\psi_k^s, \psi_t^f}\left\{ \sum_{k=0}^{\infty} \gamma^k R_k^s(\psi_k^s, \psi_{kT}^f, \nu_k, \pi_T^f) \right\},$$

where γ is the discounting factor, and Π^s and Π_T^f are the finite set of resource provisioning and task scheduling policies.

V. Offline Exact Solution

In this section, we introduce the offline exact solution for the profit maximizing problem. The system dynamics in the slow timescale is an MDP over the infinite horizon, and the reward function is profit over T time slots. The system dynamics in the fast timescale is an MDP over a finite T-horizon. The optimality equation for the two-timescale MDP [35, 38] is

$$\mathcal{P}2: V^*(\psi_k^s, \psi_{kT}^f) = \max_{\nu_k}\left\{ \max_{\pi_T^f}\left\{ R_k^s(\psi_k^s, \psi_{kT}^f, \nu_k, \pi_T^f) + \gamma \mathbb{E}_{\psi_{k+1}^s, \psi_{(k+1)T}^f}\left\{ V^*(\psi_{k+1}^s, \psi_{(k+1)T}^f) \right\} \right\} \right\}, \quad (7)$$

where $V^*(\psi_k^s, \psi_{kT}^f)$ is the optimal value of the overall discounted profit starting from state ψ_k^s and ψ_{kT}^f, and

$$\mathbb{E}_{\psi_{k+1}^s, \psi_{(k+1)T}^f}\left\{ V^*(\psi_{k+1}^s, \psi_{(k+1)T}^f) \right\} = \sum\nolimits_{\psi_{k+1}^s} \sum\nolimits_{\psi_{(k+1)T}^f} P_{\psi_{kT}^f, \psi_{(k+1)T}^f}^T(\pi_T^f, \psi_k^s, \nu_k) P^s(\psi_{k+1}^s \mid \psi_k^s, \nu_k) V^*(\psi_{k+1}^s, \psi_{(k+1)T}^f).$$

The state transition probability in the slow timescale, $P^s(\psi_{k+1}^s \mid \psi_k^s, \nu_k)$, represents the probability of the system transiting from state ψ_k^s to ψ_{k+1}^s by taking resource provisioning action ν_k. The state transition probability in the fast timescale, $P_{\psi_{kT}^f, \psi_{(k+1)T}^f}^T(\pi_T^f, \psi_k^s, \nu_k)$, represents the probability of the system transiting from state ψ_{kT}^f to $\psi_{(k+1)T}^f$ over T time slots under task scheduling policy π_T^f.

To derive the optimal policies for maximizing profit, one can in principle derive the offline optimal policies with value iteration [39]. It computes an optimal MDP policy and value function by starting with an arbitrary value function and using Eq. (8) to update value function until it converges. Then, the optimal policy can be obtained by choosing the action that maximizes the value function starting from the current state.

$$V(\psi^s, \psi^f) \leftarrow \max_{\nu}\left\{ \max_{\pi_T^f}\left\{ R^s(\psi^s, \psi^f, \nu, \pi_T^f) + \gamma \mathbb{E}_{\psi^{s'}, \psi^{f'}}\left\{ V(\psi^{s'}, \psi^{f'}) \right\} \right\} \right\}, \quad (8)$$

where $\psi^{s'}$ and $\psi^{f'}$ are the next system states. The offline exact solution using value iteration is detailed in Algorithm 1.

Algorithm 1 Offline Exact Solution for Profit Maximization

Input:
P^T, P^s: state transition probability in the two timescales.
Ψ^s, Ψ^f: state spaces of the two timescales.
Π_T^f: set of all possible T-horizon scheduling policies.
$\wedge$: action space for resource provisioning action.
$R^s(\psi^s, \psi^f, \nu, \pi_T^f)$: reward function in the slow timescale.
θ: threshold, $\theta > 0$.

Output: The optimal value function $V^*(\psi^s, \psi^f)$.

```
Set i = 0.
Set V(ψ^s, ψ^f) = 0, for all ψ^s and ψ^f.
repeat
    i = i + 1.
    for each state ψ^s ∈ Ψ^s do
        for each state ψ^f ∈ Ψ^f do
            Update V(ψ^s, ψ^f) according to Eq. (8).
        end for
    end for
until |V_i(ψ^s, ψ^f) − V_{i−1}(ψ^s, ψ^f)| ≤ θ, ∀ψ^s, ∀ψ^f
```

Theorem V.1. *In Algorithm 1, $V(\psi^s, \psi^f)$ can converge towards the optimum $V^*(\psi^s, \psi^f)$, for any ψ^s and ψ^f, in a finite number of iterations. More precisely,*

$$|V_i(\psi^s, \psi^f) - V^*(\psi^s, \psi^f)| \leq \theta, \exists i, \forall \theta, \psi^s, \psi^f. \quad (9)$$

Proof. Please refer to [39] for detailed proof. □

The system state space is large and the state transition probability can hardly be exactly measured in a real system. We will discuss the approximate solutions for deriving the task scheduling policy and resource provisioning policy.

VI. Online Approximate Solutions

In this section, we first introduce the approximation framework. Then, we present the approximate solutions for task scheduling and resource provisioning.

One difficulty to obtain the optimal solution is that the task scheduling policy affects not only the service revenue over the next T time slots but also the system states after T time slots, and thus it affects profit after the next T time slots. Because the charged price for tasks decreases with longer delays, we adopt a greedy policy that maximizes the revenue over the next T time slots. Thus, with the reward function defined as $\max(R_k^s(\cdot))$, the optimal value function for the MDP is

$$\mathcal{P}3: \hat{V}^*(\psi_k^s, \psi_{kT}^f) = \max_{\nu_k}\left\{ \max_{\pi_T^f}\left\{ R_k^s(\psi_k^s, \psi_{kT}^f, \nu_k, \pi_T^f) \right\} + \gamma \mathbb{E}_{\psi_{k+1}^s, \psi_{(k+1)T}^f}\left\{ \hat{V}^*(\psi_{k+1}^s, \psi_{(k+1)T}^f) \right\} \right\}. \quad (10)$$

The state transition probability from state ψ^f_{kT} to $\psi^f_{(k+1)T}$ is determined by the state and action in the slow timescale and the greedy task scheduling policy in the fast timescale.

A. Task Scheduling Policy in Fast Timescale

In this subsection, we derive the approximate policy for task scheduling. Our approximation is based on the greedy policy in Eq. (10), which maximizes the revenue over T time slots,

$$R^{s*}(\psi^s, \psi^f, \nu) = \max_{\pi^f_T \in \Pi^f_T} \Big\{ R^s(\psi^s, \psi^f, \nu, \pi^f_T) \Big\}, \tag{11}$$

where R^{s*} is the maximum revenue over T time slots. The greedy policy can be theoretically derived using backward induction over the finite T-horizon, however, it can hardly be obtained in practice due to large state and action spaces in the two timescales and the system dynamics over T time slots.

Our second approximation is to assume the VM number is constant before finishing the current pending tasks. This is feasible because the system scales the computing capacity at a relative low frequency. The task scheduler schedules pending tasks to maximize the revenue gained from the pending tasks,

$$\mathcal{P}4 : \max_{\ell_t} \sum_{i \in x_t} U_i(f_i), \tag{12}$$

where f_i is the completion time of task i, and ℓ_t is the task sequence. To maximize revenue, we first introduce a method for estimating the completion time of a task. Then, we present a method for deriving the task scheduling policy.

1) Task Completion Time: Videos are segmented into blocks to be transcoded on many VMs in parallel. We assume that each video block consumes F time slots, and task i consists of b_i blocks. Suppose the transcoding order of task i is o_i. Given the task sequence, we suppose the number of video blocks to be processed before finishing task i is g_i, and

$$g_i = \sum_{l' \in l} b_{l'} + b_i, \tag{13}$$

where l is the set of pending tasks ahead of task i.

For simplicity, our system does not track the transcoding progress of a video block, and we consider this information is unknown. The time consumption for data transmission is small, we ignore them in our analysis. We have the following proposition to estimate the completion time of a video block.

Proposition 1. *Suppose the system has m_k VMs. At t_0, one worker becomes idle and requests a video block. Then, the expected completion time of g_i video blocks (i.e., task i) is*

$$E\{f_i\} = t_0 + \frac{F}{m_k}(g_i - 1) + F. \tag{14}$$

Proof. Please see Appendix A for detailed proof. □

2) Task Scheduling with Adjacent Pairwise Interchange: Given a set of pending tasks x_t, we suppose task j and k are adjacent and the sequence is $(..., j, k, ...)$. Let f_j and f_k be the completion time of task j, k. The expected revenue gained from task j and k based on Eq. (1) is

$$R_{jk} = E\{\alpha^{f_j - a_j} R_j D_j\} + E\{\alpha^{f_k - a_k} R_k D_k\}. \tag{15}$$

We interchange task j and k while keeping the other tasks unchanged. The completion time of the other tasks will not be affected by the interchange. The expected revenue gained from task j and k in sequence $(..., k, j, ...)$ is

$$R_{kj} = E\{\alpha^{f'_k - a_k} R_k D_k\} + E\{\alpha^{f'_j - a_j} R_j D_j\}, \tag{16}$$

where f'_j and f'_k are the completion time of task j and k after the interchange. Based on Proposition 1, if $R_{jk} > R_{kj}$, we can deduce that

$$\frac{\alpha^{d_j - a_j} R_j D_j}{1 - \alpha^{d_j}} \geq \frac{\alpha^{d_k - a_k} R_k D_k}{1 - \alpha^{d_k}}, \tag{17}$$

where $d_j = \frac{F}{m_k} b_j$ and $d_k = \frac{F}{m_k} b_k$. d_j and d_k are derived according to Eq. (14). We have the following proposition for scheduling tasks to maximize revenue.

Proposition 2. *If a set of pending tasks x_t is scheduled in the decreasing order of P_i, the overall revenue can be maximized,*

$$P_i = \frac{\alpha^{d_i - a_i} R_i D_i}{1 - \alpha^{d_i}}, i \in x_t, where\ d_i = \frac{F}{m_k} b_i. \tag{18}$$

Proof. Please see Appendix B for detailed proof. □

The task scheduler calculates P_i for each task, and sequences tasks in the decreasing order of P_i. We illustrate the details in Algorithm 2. P_i is not time-varying, therefore, the task scheduler only requires to sequence the pending tasks when new tasks arrive or the number of VMs changes. The algorithmic complexity for sequencing tasks with Quicksort is $O(nlogn)$, where n is the number of pending tasks.

Algorithm 2 Task Scheduling Policy in Fast Timescale

Input:
x_t: current set of pending tasks.
R_i: initial marginal price for each task.
b_i: number of video blocks of each task.
m_k: current number of VMs.
Output: The task sequence ℓ_t.

```
Set t = 0
while the system is in service do
    t ← t + 1
    if New tasks come in OR VM number changes then
        for each task i ∈ x_t do
            Calculate P_i for each task according to Eq. (18)
        end for
        Sequence the tasks by the decreasing order of P_i.
    end if
end while
```

This method can also be applied to the pricing function of Eq. (1.a) for deriving the task scheduling policy. The revenue can be maximized by sequencing tasks in the decreasing order of $P_i = \beta_i / d_i$. With the pricing function of Eq. (1.b), $\mathcal{P}4$ is NP-hard. In this case, sequencing tasks in the decreasing order of $P_i = w_i / d_i$ is still shown to be a popular and effective approximate solution in the previous research [40].

Remark. Our proposed task scheduling algorithm makes decisions based on the predicted revenue of each pending task. If the predicted required computing resource of a task has

error, there must be some error of the predicted revenue of the task, thereby affecting task scheduling decisions. With random effects of prediction error, the required computing resource of a task, D_i in Eq. (1), is unknown before the task has been finished. D_i can thus be seen as a random variable. Our task scheduling algorithm maximizes the expected overall revenue of pending tasks, therefore, the solution is still optimal if the prediction is unbiased. In Fig. 5 of our experiment, it can be verified that the prediction error of the neural network method for required computing resource prediction is approximately unbiased. In this case, the random effects of the prediction error will not affect task scheduling decisions.

B. Resource Provisioning Policy in Slow Timescale

In this subsection, we introduce the method for deriving the resource provisioning policy in the slow timescale. The system dynamics in the slow timescale is an MDP with the reward defined as profit over T time slots. We can write the system dynamics in the slow timescale as

$$\mathcal{P}5: \hat{V}^*(\psi_k^s, \psi_{kT}^f) = \max_{\nu_k} \Big\{ R_k^s(\psi_k^s, \psi_{kT}^f, \nu_k, \pi_T^f) + \gamma \mathbb{E}_{\psi_{k+1}^s, \psi_{(k+1)T}^f} \big\{ \hat{V}^*(\psi_{k+1}^s, \psi_{(k+1)T}^f) \big\} \Big\}. \quad (19)$$

The resource provisioning policy determines the number of VMs to be activated or shut down based on the current system state (ψ_k^s, ψ_{kT}^f) for maximizing profit.

To derive the resource provisioning policy, we adopt *Q Learning* [41] to find the action-selection policy for the MDP in $\mathcal{P}5$. The learning procedures are as follows: at the beginning of N_k, the system observes state ψ_k^s and ψ_{kT}^f in the two timescales, and selects action ν_k according to a certain resource provisioning policy, π^s. After T time slots, the system observes the profit gained over T time slots, R_k^s, and the new system state ψ_{k+1}^s and $\psi_{(k+1)T}^f$ at time N_{k+1}. Then, the new estimated discounted overall profit starting from state (ψ_k^s, ψ_{kT}^f) by taking action ν_k can be calculated as

$$Q'((\psi_k^s, \psi_{kT}^f), \nu_k) = R_k^s(\psi_k^s, \psi_{kT}^f, \nu_k, \pi^f) + \gamma \max_{\nu_{k+1}} Q((\psi_{k+1}^s, \psi_{(k+1)T}^f), \nu_{k+1}), \quad (20)$$

where $Q((\psi_k^s, \psi_{kT}^f), \nu_k)$ is the action-value and ν_{k+1} is the optimal action that maximizes the expected overall discounted profit starting from state $(\psi_{k+1}^s, \psi_{(k+1)T}^f)$.

Action-value $Q((\psi_k^s, \psi_{kT}^f), \nu_k)$ can be updated based on the new estimation according to following equation,

$$Q((\psi_k^s, \psi_{kT}^f), \nu_k) = Q((\psi_k^s, \psi_{kT}^f), \nu_k) + \delta_k \{ Q'((\psi_k^s, \psi_{kT}^f), \nu_k) - Q((\psi_k^s, \psi_{kT}^f), \nu_k) \}, \quad (21)$$

where δ_k is the learning rate. We set the learning rate as $\frac{1}{N_\psi^\nu}$, where N_ψ^ν is the number of times that $Q((\psi_k^s, \psi_{kT}^f), \nu_k)$ has been visited. With enough times of each state-value being visited, it will converge to the optimal policy.

The above learning method requires to explore the entire system state space (Ψ^s, Ψ^f) and action space $\wedge$. However, it is hard to enumerate the entire space. To reduce the dimensionality of the space, we adopt the feature extraction method to obtain a compacted representation of the state space [42]. Let the compacted state space be Φ. The original state (ψ_k^s, ψ_{kT}^f) can be compacted as state ϕ_k, and

$$\phi_k = (\omega_{kT}, m_k, \lambda_k), \quad (22)$$

where ω_{kT} is the overall revenue of the pending tasks at time kT, m_k is the number of VMs, λ_k is the task arrival rate. We replace (ψ_k^s, ψ_{kT}^f) with ϕ_k in Eq. (20) and (21).

During the learning processes, if the resource manager always chooses the action which maximizes $Q(\phi_k, \nu_k)$, it may converge to a local optimum and fail to find the optimal policy. To address it, we adopt the ε-$greedy$ method to balance the exploring and exploiting during learning. Specifically, the resource manager chooses an action randomly with the probability of ε for exploration, or chooses the action which maximizes $Q(\phi_k, \nu_k)$ with the probability of $1 - \varepsilon$ for exploitation. Our method for learning the resource provisioning policy in the slow timescale is illustrated in Algorithm 3.

Algorithm 3 Resource Provisioning Policy in Slow Timescale

Input:
Initialize $Q(\phi, \nu) = C$, $\forall \phi$, ν.
Task scheduling policy π_T^f.
Set the maximum number of loops, M.
Set k = 0.

Output: The optimal action-value $Q^*(\phi, \nu)$.

1: Obtain the current system state ϕ_k at the beginning of N_k.
2: **repeat**
3: Select ν_k based on ϕ_k and $Q(\phi, \nu)$ using ε-$greedy$.
4: Take action ν_k, observe the overall profit R_k^s over T time slots in N_k under task scheduling policy π^f.
5: Observe system state ϕ_{k+1} at the beginning of N_{k+1}.
6: Update action-value $Q(\phi_k, \nu_k)$ according to Eq. (21)
7: $k \leftarrow k + 1$
8: **until** $k < M$

Theorem VI.1. *$Q(\phi, \nu)$ converges to the optimum, $Q^*(\phi, \nu)$, for any ϕ and ν, if all actions are repeatedly sampled in all states, i.e., $Q_k(\phi, \nu) \to Q^*(\phi, \nu)$ as $k \to \infty$, $\forall \phi, \nu$.*

Proof. Please refer to [43] for detailed proof. □

The above algorithm can be applied online, however, it may incur large loss at the initial stage. We can use the system running trajectory or simulation to train it offline and then apply it online to adapt to a real system.

VII. Performance Evaluation

In this section, we first present our system implementation and experiment settings. Then, we evaluate the performance of our method through extensive experiments.

A. System Implementation

We design our cloud transcoding system, *Morph*, following the master-worker paradigm. The source code is available in Github [44]. The main system components are illustrated in Fig. 4. Users can access the system via Command Line

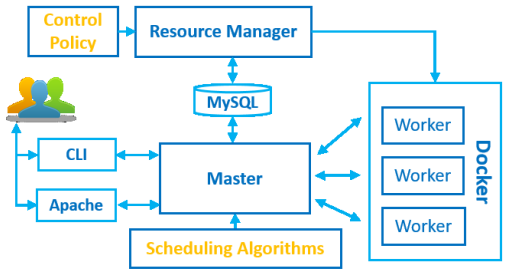
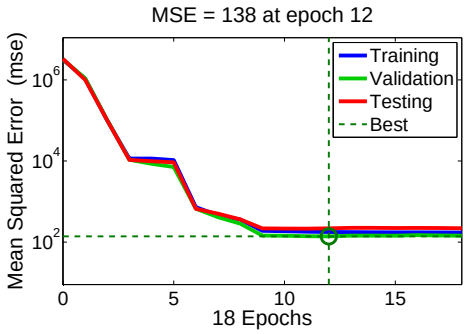
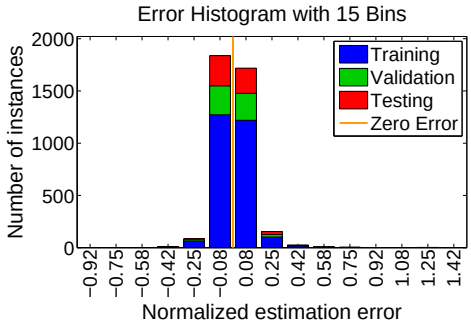


Fig. 4. The main components of our cloud transcoding system.



(a) MSE of neural network method



(b) Error histogram of neural network method

Fig. 5. The performance of the neural network method.

Interface (CLI) or web portal. The HTTP requests from users are processed by Apache web server. We use MySQL to store system information. The resource provisioning and task scheduling algorithms are implemented in a library. We build the cloud environment with Docker [45], which is a software containerization platform. Our system can dynamically control the number of active computing instances by starting or stopping Docker containers. We use FFmpeg [46] for segmenting videos and transcoding video blocks. The details of FFmpeg commands for video segmenting, transcoding, and concentration are illustrated in Appendix C. The system consists of a master node, a resource manager node, and many transcoding worker nodes. Each node runs on a virtual machine. The main functionalities of each node are shown as follows. More details of the system design are presented on our GitHub page [44].

Master Node: The service interface module and task scheduling module are implemented in the master node. The master node processes user requests and schedules transcoding tasks. For each transcoding task, it segments the video file into blocks and sends the video blocks to the transcoding workers when receiving requests. The master node will concentrate the transcoded video blocks of a task into one video file when the task has been completed. The master node also maintains a timer for each video block that has been dispatched to the transcoding workers. A video block will be re-dispatched to another transcoding worker if the master node does not receive the transcoded video block within a prescribed deadline. A transcoding task will be reported as a failure if a maximum number (e.g., 3 times) of retries for a video block all fail.

Transcoding Worker Node: The system maintains many virtual machines in the cloud for video transcoding. Each virtual machine that is dedicated to video transcoding runs a transcoding worker. The transcoding workers request video blocks from the master node and transcode the video blocks into the target representations. When the transcoding for a video block is completed, the transcoding worker will send the transcoded video block to the master node.

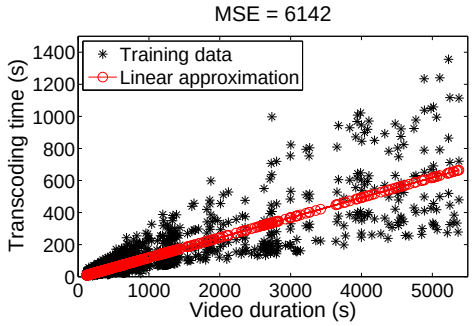
Resource Manager Node: The resource manager is deployed on a virtual machine. It dynamically controls the number of active transcoding workers in the system according to the system workloads and the resource provisioning policy.

B. Experiment Settings

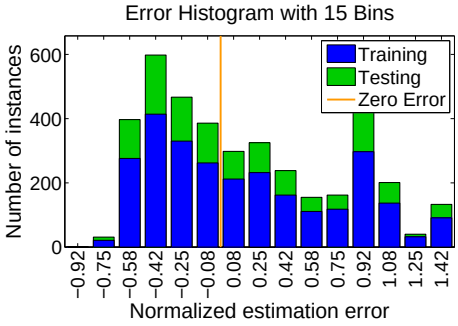
We adopt Amazon EC2 pricing mechanism to calculate computing cost. The cost for a computing instance is \$0.252 per hour. Each container has 4 CPU cores, the CPU frequency is 2.10 GHz, and the memory size is 2GB. The servers are connected with Gigabit Ethernet. The target video resolution is randomly selected from 854x480, 640x360, and 426x240. The service has three priority levels, namely, Level I, II, and III. The initial marginal price R_i is \$0.018 per minute for level I, \$0.012 per minute for level II, and \$0.006 per minute for level III. The discounting factor α is 0.995 per 5 seconds. The consumed computing time of each video block is 180 seconds. The task scheduler schedules tasks every 5 seconds, and the resource manager scales the computing capacity every one hour. Tasks arrive according to the poisson process with different rates, and the priority level is randomly selected. To compare the performance of different schemes, we first generate a sequence of tasks and record the task information. We use the same sequence of tasks for evaluating the performance of different schemes to ensure the fairness.

C. Computing Resource Demand Estimation Accuracy

To evaluate the estimation accuracy of the required computing resource of a task, we measure the time of transcoding 2020 videos into three target resolutions, namely, 854x480, 640x360, 426x240. We obtain 3850 instances of the measured transcoding time. We select 75% of the data for training the neural network, 15% for model validation, and 15% for testing. The hidden layer of the neural network consists of 20 neurons. We adopt the Levenberg-Marquardt backpropagation algorithm for training. The input feature vector of the neural network consists of the bitrate, resolution, frame rate, duration of the original video, and the resolution of the target representation. We use the product of the width and height of the

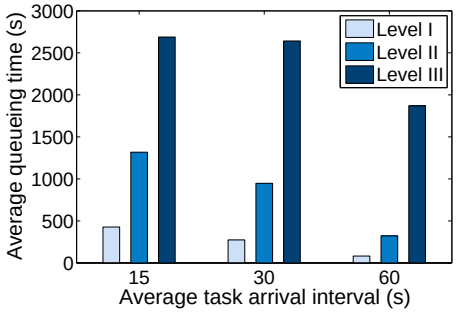


(a) Linear approximation method

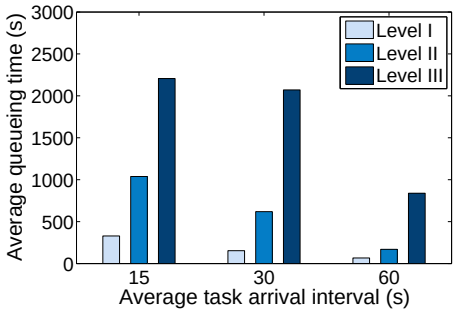


(b) Error histogram of linear approximation method

Fig. 6. The performance of the linear approximation method.



(a) 12 VMs



(b) 16 VMs

Fig. 7. Average queuing time of the three priority levels under different task arrival intervals and different system scales.

video resolution as one input. We compare the neural network method with the linear approximation method, which estimates transcoding time as a linear function of video duration. We normalize estimation error using the following equation

$$Normalized\ Error = \frac{Estimated\ Time - Real\ Time}{Real\ Time}.$$

We illustrate the Mean Squared Error (MSE) of the neural network method at different training epochs in Fig. 5(a). The MSE converges with more training epochs. The best validation performance is obtained at the 12th epoch, and the MSE is 138. The error histogram of the neural network method is illustrated in Fig. 5(b). The normalized estimation error of the testing instances are mostly within the range from -0.08 to 0.08. The relation between video duration and transcoding time is illustrated in Fig. 6(a). The MSE of the linear approximation method is 6142, much larger than that of the neural network method. The error histogram of the linear approximation method is illustrated in Fig. 6(b), the normalized estimation error of the testing instances ranges from -0.58 to 1.42. The comparisons verify that the neural network method can estimate transcoding time very precisely.

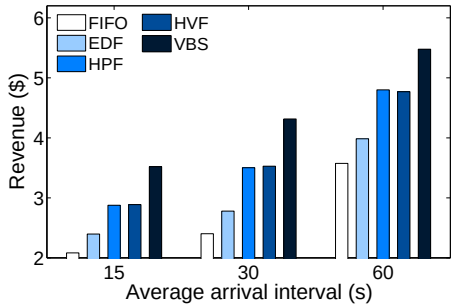
D. Average Queueing Time for Different Priority Levels

The service has three priority levels. To compare the service of different priority levels, we evaluate the average queueing time (i.e., waiting time in queue before transcoding) for the three priority levels under our task scheduling scheme. We submit 50 tasks to the system according to the Poisson process with different task arrival intervals and different number of VMs. We measure the average queuing time for the tasks. The results are demonstrated in Fig. 7. It can be observed that the average queueing time of priority level I is the smallest and priority level III is the largest under different system scales and task arrival rates. This verifies that users can get a higher quality of service by selecting a higher priority level, and the average queuing time will be smaller. Users who require low delays can select priority level I while paying a higher price. Users who desire low price can select priority level III, but the task processing delays would be longer.

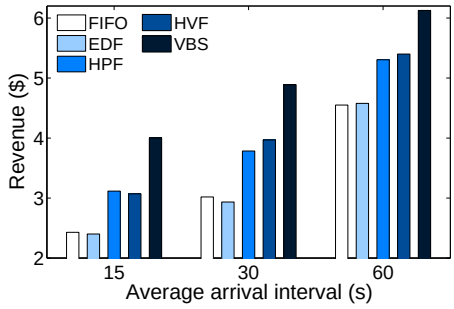
E. Comparison of Revenue with Other Schemes

We compare our task scheduling scheme with the following baselines. 1) First In First Out (FIFO): tasks are scheduled following their arrival orders. 2) Earliest Deadline First (EDF): the task with the earliest deadline will be selected first. We set the deadline of a task as three times of the estimated transcoding time since its submission. 3) Highest Priority First (HPF): the task with the highest priority will be selected first. 4) Highest Value First (HVF): the task with the highest current price will be selected first. The current price of a task is calculated according to Eq. (1). Our task scheduling scheme is referred as value-based scheduling scheme (VBS).

We compare the revenue gained from 50 tasks under different task scheduling schemes. We measure the revenue gained from the tasks under different task arrival intervals and different system scales. The results are illustrated in Fig. 8. We can observe the revenue with our VBS is larger than the baselines. Because FIFO and EDF only consider the arrival order or the deadline, and do not consider the possible revenue



(a) 12 VMs



(b) 16 VMs

Fig. 8. The revenue under different scheduling schemes and system scales.

Scenario	Arrival Rate per Minute	Valuation of Pending Tasks	Optimal Number of VM instances
1	0.1	0.0	3
3	0.1	0.2	4
9	0.1	0.8	5
12	0.2	0.0	6
17	0.2	0.5	7
23	0.3	0.0	8
27	0.3	0.4	10
...	....	...	...

Fig. 9. The relation among task arrival rate, valuation of the pending tasks, and the optimal number of VMs.

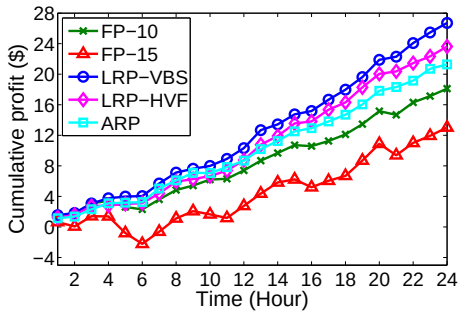


Fig. 10. The cumulative profit with different methods over 24 hours.

that can be gained from the tasks. While HVF and HPF select the task with the current highest priority or price to perform, but they do not consider factors such as the decreasing of the price with more delays or the number of available VMs.

The gained revenue from the tasks are higher with larger task arrival intervals or more VMs. With larger task arrival interval, there will be less computing resource competition among pending tasks, it will incur less processing delays and the revenue will be higher. With more VMs, the tasks can be completed faster, there will be less decay of revenue. However, larger task arrival interval means less revenue, and provisioning more VMs incurs more cost. Hence, the resource manager must determine the optimal number of provisioned VMs for maximizing service profit.

F. Comparison of Service Profit with Other Schemes

We measure the service profit under a real workload trace. The workload trace captures video requests to a CDN node. We extract the user requests in the trace as the transcoding requests to our system. We divide the time of each day into 24 hours and average the request number during one hour (e.g., 9:00 AM - 10:00 AM) of the days as the average task arrival rate in this hour of a day. We scale down the average request rate to 0.1-0.7 request per minute. We use the scaled numbers of requests in each hour as task arrival rates to train the resource provisioning policy. We illustrate our obtained Learning-based Resource Provisioning (LRP) policy under the task scheduling scheme VBS in Fig. 9. The result shows the relation among the valuation of pending tasks, task arrival rate, and the optimal number of VMs to maximize profit. Intuitively, it requires more VMs with larger task arrival rate. Meanwhile, with larger valuation of pending tasks, it requires more VMs because more revenue can be obtained.

We measure the profit in a real environment over 24 hours with the combinations of different resource provisioning policies and task scheduling policies. We compare the service profit under our proposed Learning-based Resource Provisioning policy and Value-based Task Scheduling (LRP-VBS) policy with the following methods: 1) The Fixed Policy (FP) which runs a fixed number of VMs with the task scheduling policy of VBS. We select two representative numbers of VMs to illustrate, namely, 10 VMs and 15 VMs. 2) The Arrival Rate based Policy (ARP), the number of provisioned VMs is proportional to the task arrival rate. This method estimates the required number of VMs based on task arrival rates. We set the number of provisioned VMs as 30 times of the task arrival rate per minute. 3) The learning-based resource provisioning policy combined with the task scheduling policy of HVF (LRP-HVF).

We illustrate the cumulative profit with different methods over the 24 hours in Fig. 10. We can observe that the cumulative service profit with LRP-VBS is larger than the baseline methods. In Fig. 11, LRP can effectively control the number of provisioned VMs with the task scheduling policy of HVF or VBS. Meanwhile, the task scheduling policy also affects the resource provisioning policy. LRP-VBS can gain a higher profit than LRP-HVF because VBS can outperform HVF for task scheduling. In constrast, FP wastes much computing resource when workloads are low and cannot meet performance requirements when workloads exceed the current computing capacity. Thus, the service profit is lower.

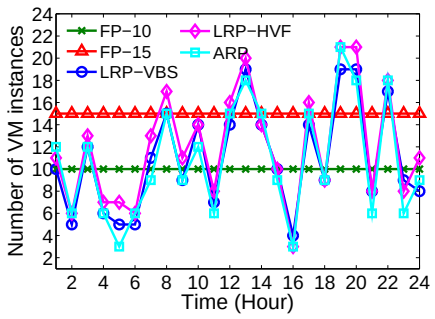


Fig. 11. The number of provisioned VMs in each hour.

ARP can dynamically control the number of provisioned VMs according to task arrival rates, however, it is hard for this method to model the decay of revenue with processing delays. The service revenue is also affected by the task scheduling policy, which will in turn affect the resource provisioning policy. However, ARP does not consider the task scheduling scheme. Therefore, ARP cannot effectively determine the optimal number of VMs for maximizing profit. LRP is suitable for such hard-to-model problems, and it can obtain the optimal policy by learning in a dynamic system.

VIII. Conclusions

Cloud transcoding services introduce many challenges regarding performance and cost. On one hand, users' desire for minimizing processing delays requires abundant resources; on the other hand, service providers are keen on maximizing financial profit. We study how to provision resources and schedule tasks to meet users' performance requirements while maximizing the service provider's profit. We propose a two-timescale stochastic optimization framework for profit maximization while meeting performance requirements by jointly provisioning resources and scheduling tasks under a hierarchical control architecture. We derive the offline exact solution, and design some approximate solutions for task scheduling and resource provisioning. We implement an open source cloud transcoding system and evaluate the performance of our method in a real environment. The results demonstrate our method can reduce resource consumption while achieving a higher profit compared with baseline schemes. In our future research, we will consider the dynamic resource provisioning problem for transcoding videos with different characteristics, e.g., advertisements, live sports, news, or TV shows. Another promising direction in this field is to thoroughly evaluate the performance gaps among different algorithms under different system settings and workloads in a real environment.

Appendix A
Proof of Proposition 1

We denote the remaining time to complete transcoding the current video block on VM i as y_i. We assume that at t_0, the first worker becomes idle and requests a video block. The waiting time for the next video block to be requested is

$$Y = \min\{y_1 = F, y_2, y_3, ..., y_{m_k}\} = \min\{y_2, y_3, ..., y_{m_k}\},$$

where $y_2, y_3, ..., y_{m_k}$ are unknown and randomly within $[0,\ F]$. The cumulative distribution function (CDF) of the remaining time to complete transcoding the current video block on VM i can be given as

$$F_i(t) = P(y_i \le t) = \frac{t}{F}, i \in [2, m_k], t \in [0, F]. \tag{23}$$

Because video blocks are transcoded independently on the VMs, the CDF of Y can be given as

$$\begin{aligned} F_Y(t) &= P(0 \le Y \le t) = 1 - P(y_2 > t, y_3 > t, ..., y_{m_k} > t) \\ &= 1 - P(y_2 > t)P(y_3 > t)...P(y_{m_k} > t) \end{aligned}$$

We denoted the PDF of Y as $f_Y(t)$. Hence, the expected waiting time for the next block to be requested is

$$E\{Y\} = \int_0^F t f_Y(t) dt = \frac{F}{m_k}.$$

We can deduce that the total waiting time for the g_i-th video block to be requested by a worker is

$$T_{g_i} = \frac{F}{m_k}(g_i - 1). \tag{24}$$

The estimated completion time of the g_i-th video block is the sum of the waiting time and transcoding time, therefore, it is in accordance with Proposition 1.

Appendix B
Proof of Proposition 2

We assume that the current set of pending tasks has been sequenced by the decreasing order of P_i. The transcoding order of task i is o_i. If we move task i from o_i to o_i', it can be done by iteratively interchanging task i with its adjacent task until it reaches o_i'. Since the tasks have been sequenced in the decreasing order of P_i, each interchanging will incur a loss on the revenue according to Eq. (17). Hence, it's optimal to schedule tasks in the decreasing order of P_i for maximizing the revenue of the pending tasks.

Appendix C
Transcoding Operations

1. Segment a video into blocks:

```
$ ffmpeg -i PV.mp4 -f segment
         -segment_time 120
         -c copy -segment_list PV.lst
          PV_%03d_.mp4
```

2. Transcode a video block into the target resolution:

```
$ ffmpeg -i id.mp4
         -c:v libx264 -c:a aac
         -s 854x480 id_854.mp4
```

3. Concentrate video blocks into one video file:

```
$ ffmpeg -f concat -i PV.lst
         -c copy PV_854x480.mp4
```

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their insightful comments. The work was supported in part by Singapore MOE tier-1 fund (RG 17/14), SMART Innovation Grant, and EIRP02 Grant from Singapore EMA.

REFERENCES

[1] T. Stockhammer, "Dynamic adaptive streaming over http: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 133–144.

[2] H. Hu, Y. Wen, H. Luan, T.-S. Chua, and X. Li, "Toward multiscreen social tv with geolocation-aware social sense," *IEEE MultiMedia*, vol. 21, no. 3, pp. 10–19, 2014.

[3] Y. Wen, X. Zhu, J. Rodrigues, and C. Chen, "Cloud mobile media: Reflections and outlook," *Multimedia, IEEE Transactions on*, vol. 16, no. 4, pp. 885–902, 2014.

[4] J. Si, S. Ma, X. Zhang, and W. Gao, "Adaptive rate control for high efficiency video coding," in *Visual Communications and Image Processing (VCIP), 2012 IEEE*. IEEE, 2012, pp. 1–6.

[5] G. Gao, W. Zhang, Y. Wen, Z. Wang, and W. Zhu, "Towards cost-efficient video transcoding in media cloud: Insights learned from user viewing patterns," *Multimedia, IEEE Transactions on*, vol. 17, no. 8, pp. 1286–1296, 2015.

[6] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with h. 264/avc: tools, performance, and complexity," *Circuits and Systems magazine, IEEE*, vol. 4, no. 1, pp. 7–28, 2004.

[7] J. Careless, "Cloud video encoding: When to go online and when to stay in-house," *Streaming Media*, 2012.

[8] J. Tang, W. P. Tay, and Y. Wen, "Dynamic request redirection and elastic service scaling in cloud-centric media networks," *Multimedia, IEEE Transactions on*, vol. 16, no. 5, pp. 1434–1445, 2014.

[9] W. Zhang, Y. Wen, J. Cai, and D. O. Wu, "Toward transcoding as a service in a multimedia cloud: energy-efficient job-dispatching algorithm," *Vehicular Technology, IEEE Transactions on*, vol. 63, no. 5, pp. 2002–2012, 2014.

[10] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 5, pp. 1378–1391, 2013.

[11] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, "Data centers power reduction: A two time scale approach for delay tolerant workloads," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 1431–1439.

[12] W. Wang, P. Zhang, T. Lan, and V. Aggarwal, "Datacenter net profit optimization with deadline dependent pricing," in *Information Sciences and Systems (CISS), 2012 46th Annual Conference on*. IEEE, 2012, pp. 1–6.

[13] J. Cao, K. Hwang, K. Li, and A. Y. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 1087–1096, 2013.

[14] K. Tsakalozos, H. Kllapi, E. Sitaridi, M. Roussopoulos, D. Paparas, and A. Delis, "Flexible use of cloud resources through profit maximization and price discrimination," in *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 2011, pp. 75–86.

[15] J. Zhao, H. Li, C. Wu, Z. Li, Z. Zhang, and F. C. Lau, "Dynamic pricing and profit maximization for the cloud with geo-distributed data centers," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 118–126.

[16] G. Gao and Y. Wen, "Morph: A fast and scalable cloud transcoding system," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 1160–1163.

[17] G. Gao, Y. Wen, and C. Westphal, "Resource provisioning and profit maximization for transcoding in information centric networking," *2016 Infocom Workshop on Multimedia Streaming in Information/Content-Centric Networks (MuSIC)*, 2016.

[18] Z. Wang, L. Sun, C. Wu, W. Zhu, and S. Yang, "Joint online transcoding and geo-distributed delivery for dynamic adaptive streaming," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 91–99.

[19] F. Jokhio, A. Ashraf, S. Lafond, I. Porres, and J. Lilius, "Prediction-based dynamic resource allocation for video transcoding in cloud computing," in *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*. IEEE, 2013.

[20] M. Song, Y. Lee, and J. Park, "Scheduling a video transcoding server to save energy," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, no. 2s, pp. 45:1–45:23, Feb. 2015. [Online]. Available: http://doi.acm.org/10.1145/2700282

[21] S. Lin, X. Zhang, Q. Yu, H. Qi, and S. Ma, "Parallelizing video transcoding with load balancing on cloud computing," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*. IEEE, 2013, pp. 2864–2867.

[22] H. Ma, B. Seo, and R. Zimmermann, "Dynamic scheduling on video transcoding for mpeg dash in the cloud environment," in *Proceedings of the 5th ACM Multimedia Systems Conference*. ACM, 2014.

[23] Z. Li, Y. Huang, G. Liu, F. Wang, Z.-L. Zhang, and Y. Dai, "Cloud transcoder: Bridging the format and resolution gap between internet videos and mobile devices," in *NOSSDAV '12*.

[24] C. Timmerer, D. Weinberger, M. Smole, R. Grandl, C. Müller, and S. Lederer, "Cloud-based transcoding and adaptive video streaming-as-a-service," *E-LETTER*.

[25] Y. Wu, Z. Zhang, C. Wu, Z. Li, and F. Lau, "Cloudmov: cloud-based mobile social tv," *Multimedia, IEEE Transactions on*, vol. 15, no. 4, pp. 821–832, 2013.

[26] W. Xiao, W. Bao, X. Zhu, C. Wang, L. Chen, and L. T. Yang, "Dynamic request redirection and resource provisioning for cloud-based video services under heterogeneous environment."

[27] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *Cloud Computing, IEEE Transactions on*, vol. 2, no. 1, pp. 14–28, 2014.

[28] Y. Ran, J. Yang, S. Zhang, and H. Xi, "Dynamic iaas computing resource provisioning strategy with qos constraint," *Services Computing, IEEE Transactions on*, 2015.

[29] Q. Duan, "Modeling and performance analysis for composite network–compute service provisioning in software-defined cloud environments," *Digital Communications and Networks*, vol. 1, no. 3, pp. 181–190, 2015.

[30] M. Ghamkhari and H. Mohsenian-Rad, "Energy and performance management of green data centers: A profit maximization approach," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 1017–1025, 2013.

[31] I.-H. Hou and P. Kumar, "Utility maximization for delay constrained qos in wireless," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[32] D. Niyato and E. Hossain, "Wireless broadband access: Wimax and beyond-integration of wimax and wifi: Optimal pricing for bandwidth sharing," *IEEE communications Magazine*, vol. 45, no. 5, pp. 140–146, 2007.

[33] S. Wang and C. Wang, "Joint optimization of spectrum and energy efficiency in cognitive radio networks," *Digital Communications and Networks*, vol. 1, no. 3, pp. 161–170, 2015.

[34] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *Signal Processing Magazine, IEEE*, vol. 20, no. 2, pp. 18–29, 2003.

[35] H. S. Chang, P. J. Fard, S. Marcus, M. Shayman *et al.*, "Multitime scale markov decision processes," *Automatic Control, IEEE Transactions on*, vol. 48, no. 6, pp. 976–987, 2003.

[36] C. Wernz, "Multi-time-scale markov decision processes for organizational decision-making," *EURO Journal on Decision Processes*, vol. 1, no. 3-4, pp. 299–324, 2013.

[37] R. E. Parr, "Hierarchical control and learning for markov decision processes," Ph.D. dissertation, Citeseer, 1998.

[38] M. S. Mahmoud, "Multilevel systems control and applications: A survey," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 7, no. 3, pp. 125–143, 1977.

[39] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[40] M. L. Pinedo, *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media, 2012.

[41] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[42] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: an overview," in *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, vol. 1. IEEE, 1995, pp. 560–564.

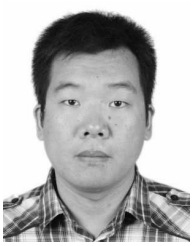
[43] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[44] "Morph: Cloud Video Transcoding," https://github.com/cap-ntu/Morph.

[45] "Docker: software containerization platform," https://www.docker.com/.

[46] "FFmpeg: a complete, cross-platform solution to record, convert and stream audio and video," https://ffmpeg.org/.

Guanyu Gao received his Master's degree from University of Science and Technology of China (USTC) in 2012 and Bachelor's degree from University of Electronic Science and Technology of China (UESTC) in 2009. He is currently a Ph.D. student in the Interdisciplinary Graduate School at Nanyang Technological University (NTU). His research interests include multimedia communication, cloud computing, and video processing.



Han Hu received the B.S. and Ph.D. degrees from University of Science and Technology of China, Hefei, China, in 2007 and 2012, respectively. He is a Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include social media analysis and distribution, big data analytics, multimedia communication, and green network.

Yonggang Wen (S99-M08-SM14) is an associate professor with School of Computer Science and Engineering at Nanyang Technological University, Singapore. He received his PhD degree in Electrical Engineering and Computer Science (minor in Western Literature) from Massachusetts Institute of Technology (MIT), Cambridge, USA, in 2008. Previously he has worked in Cisco to lead product development in content delivery network, which had a revenue impact of 3 Billion US dollars globally. Dr. Wen has published over 140 papers in top journals and prestigious conferences. His work in Multi-Screen Cloud Social TV has been featured by global media (more than 1600 news articles from over 29 countries) and received ASEAN ICT Award 2013 (Gold Medal). His work on Cloud3DView, as the only academia entry, has won the Data Centre Dynamics Awards 2015 C APAC. He is a co-recipient of 2015 IEEE Multimedia Best Paper Award, and a co-recipient of Best Paper Awards at EAI/ICST Chinacom 2015, IEEE WCSP 2014, IEEE Globecom 2013 and IEEE EUC 2012. He serves on editorial boards for IEEE Transactions on Circuits and Systems for Video Technology, IEEE Wireless Communication Magazine, IEEE Communications Survey & Tutorials, IEEE Transactions on Multimedia, IEEE Transactions on Signal and Information Processing over Networks, IEEE Access Journal and Elsevier Ad Hoc Networks, and was elected as the Chair for IEEE ComSoc Multimedia Communication Technical Committee (2014-2016). His research interests include cloud computing, green data center, big data analytics, multimedia network and mobile computing.

Cedric Westphal (SM08) received the M.S.E.E. degree from Ecole Centrale Paris, Paris, France, in 1995, and the M.S. and Ph.D. degrees in electrical engineering from University of California at Los Angeles, Los Angeles, CA, USA, in 1995 and 2000, respectively. He was with Nokia Research Center, Sunnyvale, CA, USA, from 2000 to 2006. He was with the Networking Architecture Group, DOCOMO Innovations Inc., Palo Alto, CA, USA, from 2007 to 2011, where he had been involved in several topics, all related to next-generation network architectures, including scalable routing, network virtualization, and reliability, using social networks for traffic offloading. He has been an Adjunct Assistant Professor with University of California at Santa Cruz, Santa Cruz, CA, USA, since 2009. He is currently a Principal Research Architect with Huawei Innovations Center, Santa Clara, CA, USA, where he is involved in future network architectures, both for wired and wireless networks. He has co-authored over 50 journal and conference papers, including several best paper awards, and holds 20 patents. His research interests include information centric networks. Dr. Westphal has been an Area Editor of ACM Transactions on Networking and IEEE TRANSACTIONS ON NETWORKING since 2009, an Assistant Editor of Computer Networks (Elsevier) journal, and a Guest Editor of Ad Hoc Networks journal. He was a reviewer for the NSF, GENI, the EU FP7, and other funding agencies.