Adaptive Configuration of Cloud Video Transcoding

Ming Yang, Jianfei Cai, Weiwen Zhang, Yonggang Wen
Nanyang Technological University, Singapore

Chuan Heng Foh
University of Surrey, Guildford

***Abstract*—Cloud computing is emerging as a new paradigm which enables big data computing, including high quality media processing. However, considering the media dynamics on resource consumption and the QoS criteria, dynamically providing the cloud computing resource to meet the QoS requirements of media processing is not easy. The current cloud computing infrastructure usually employs auto-scaling to dynamically adjust the computing resource allocation, which is typically performed at relatively long time scale and cannot adapt to the dynamic changes of video arrivals or content changes at relatively short time scale. In this paper, we propose to adaptively configure the video transcoding mode to deal with the short-term transcoding QoS and computing resource mismatch problem. We formulate the problem as the one to minimize the output bit-rate with the queue stability constraint, for which we use the Lyapunov optimization framework to solve it. Simulation results show that, compared with the static configuration strategy, the proposed adaptive method achieves smooth transcoding QoS degradation when system load becomes heavier and much better transcoding delay performance.**

I. Introduction

Video transcoding is popular due to the proliferation of video content and the heterogeneity of video format, devices, network conditions and user preference. Since video transcoding is a computation-intensive process, the demands on high performance computing keep increasing with the growing magnitude of concurrently arrived transcoding jobs. By leveraging the powerful and elastic cloud computing, cloud based transcoding, such as the Amazon Elastic Transcoder and Zencoder, is more convenient and economical, compared with maintaining and upgrading private servers. In cloud based transcoding, by renting different numbers of virtual machines (VMs) from the cloud infrastructure provider, the system computing capacity can be scaled up and down quickly to meet dynamic transcoding service requests.

In literature, a few cloud-assisted video coding/transcoding systems have been developed, such as map-reduce framework based multimedia transcoding platform [1] and social-aware video transcoding/prefetching system [2]. However, the major challenge for cloud-based video trancoding system is how to provide QoS in terms of video quality, compression ratio and transcoding delay with respect to computing resource consumption. It is unclear about the relationship between computing resource consumption and the transcoding QoS. There are some previous works on the complexity-rate-distortion analysis [3] of video coding, which try to derive some general models. But the models are limited to some assumptions and video content prior knowledge and it is challenging to adapt them to the cloud transcoding scenario.

There are some cloud trancoding works [4], [5] discussing the cloud resource consumption and the transcoding QoS. One category of research is on the admission control of the transcoding job requests so as to guarantee QoS of the running jobs under limited computing resource. For example, a stream-based admission control and scheduling approach (SBACS) was proposed in [6], which uses the queue waiting time of transcoding servers to make admission control decisions, including admit, defer, or reject. Another class of works is to schedule the computing resource provisioning to maximize the revenue for the transcoding service provider. For example, Song et al. [7] proposed a profit-aware dynamic biding algorithm to maximize the time-average profit of the cloud service broker, which is self-adaptive and requires no a priori statistical knowledge of the job distribution and arrival. Another category of works focuses on the transcoding job scheduling. For instance, Ma et al. [8] proposed a dynamic scheduling method on video transcoding for MPEG-DASH in a cloud scenario. The designed scheduler monitors the workload on each machine and dispatches high-priority jobs to the fast processing machines while low-priority jobs are dispatched to slow processing machines. The algorithm also adjusts the video transcoding mode according to the system load. All these existing works mainly focus on either the system architecture design or heuristic job scheduling design to provide QoS and they typically require the prior knowledge of video workload sizes in the scheduling.

In cloud video transcoding, at a particular period only a certain amount of computing resource is allocated to a transcoding job. Since the video content is quite diverse, dynamic computing resource is needed at different time. The current cloud computing infrastructure typically employs auto-scaling to dynamically adjust the computing resource allocation. However, the auto-scaling is typically performed at relatively long time scale, say hours. The computing resource cannot adapt to the dynamic changes of video arrivals or content changes at relatively short time scale, say within one hour.

Thus, in this paper, we propose to adaptively configure the video transcoding mode to deal with the short-term transcoding QoS and computing resource mismatch problem. The transcoding mode we refer to here is the configuration that controls the transcoding complexity. In particular, a simple transcoding configuration results in fast processing speed but low compression ratio, while a complex configuration leads to high compression ratio but long transcoding delay. So, essentially the problem becomes how to choose the optimal transcoding configuration at different time so as to trade off between the transcoding delay and the compression ratio under a given cloud computing resource. We formulate such a problem as the one to minimize the output bit-rate with the queue stability constraint, for which we use the Lyapunov optimization framework [9] to solve it. Simulation results

978-1-4799-8391-9/15/$31.00 ©2015 IEEE

show that, compared with the static configuration strategy, the proposed adaptive method achieves much better output bit-rate vs transcoding delay results.

II. System Models

Fig. 1 shows the diagram of the studied cloud-based video transcoding system. In our scenario, a certain duration of video segment files from content provider (such as Stream-a and Stream-b) are uploaded to the cloud transcoding system. Upon arrival, each job is dispatched to a virtual machine (VM) within cloud to be processed. The load balancer in job dispatcher is to balance the load among different VMs. After allocated to a VM, a job will be buffered to the affiliated first-come-first-served (FCFS) queue to wait for transcoding service if VM is busy. Once a VM is free to process the next job, the first job in its queue will start the transcoding process with a selected configuration.

In particular, for simplicity, we analyze the system using a discrete time slot based model, which supposes the job arrival event is triggered at the beginning of each time slot and the duration of the slot is short. The admitted video transcoding jobs are heterogeneous in terms of the computing resource consumption and the compression performance. To keep the load balance among the VMs, the dispatching follows the shortest queue backlog first policy. The queue backlog is defined as the accumulated computing time required. In each discrete time slot, we further assume that there is at most one video segment arrival event in each queue.

The transcoding preset configuration we consider here determines the video transcoding complexity including the number of B frames, the number of maximum reference pictures, the subpixel estimation complexity, etc. For the fastest speed configuration, the bit-rate of the generated video is expected to be largest for the same quantization parameter (QP), since the corresponding coarse coding algorithm can only exploit limited temporary and spatial redundancy. We fix the QP value for all the transcoding configurations so that the video quality will be the same for different configurations. This is reasonable since users typically prefer high visual quality. In this way, the transcoding QoS mainly consists of compression ratio (or output bit-rate) and transcoding delay.

III. Problem Formulation and Solution

Let $\overline{\mathbf{D}}$ denote the time-average bit-rate of the transcoding output:

$$\overline{\mathbf{D}} \triangleq \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} D(t), \tag{1}$$

where $D(t)$ is the output bit-rate at time slot t. Let $\overline{\mathbf{Q}}$ denote the time-average queue backlog:

$$\overline{\mathbf{Q}} \triangleq \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{|Q(t)|\}. \tag{2}$$

where $Q(t)$ is the queue backlog at time slot t. We formulation the transcoding configuration task as a minimization problem:

$$\min_{a(t)} \quad \overline{\mathbf{D}}, \tag{3}$$

$$\text{s.t.} \quad \overline{\mathbf{Q}} < \infty, \tag{4}$$

$$a(t) \in \mathcal{A}, \ \forall t, \tag{5}$$

where $a(t)$ is the transcoding preset at time slot t that needs to be determined, and the variable $a(t)$ is chosen from $\mathcal{A} = \{0, 1, ..., L-1\}$, which represents the alternative L types of transcoding presets. Note that (4) is to ensure the queue stability so as to avoid unbounded job completion delay.

The problem defined in (3) can be handled by the Lyapunov optimization framework [9]. In particular, by leveraging the Lyapunov framework, the constraint (4) is converted to the queuing style. The workload backlog $Q(t)$ is updated when new job arrives or the queuing job is executed, i.e.

$$Q(t+1) = max[Q(t) - c_a(t), 0] + \lambda(t), \tag{6}$$

where $\lambda(t)$ is the workload size of the new job arrival in the time slot, and $c_{a(t)}(t)$ refers to the workload executed in the time slot.

The min drift-plus-penalty approach [9] is used to solve the problem, where we define the quadratic Lyapunov function as

$$L(Q(t)) \triangleq \frac{1}{2}Q(t)^2, \tag{7}$$

and the *conditional Lyapunov drift* to describe the variation of queue $\mathbf{Q}$ backlog,

$$\Delta(Q(t)) \triangleq \mathbb{E}\{L(Q(t+1)) - L(Q(t))|Q(t)\}. \tag{8}$$

A small value of $L(Q(t))$ indicates that the queue backlog is small. The drift $\Delta(Q(t))$ is the expected change in the Lyapunov function over one time slot, given the current queue backlog state $Q(t)$ in time slot t.

To exploit the tradeoff between the queue backlog and the compression performance, we follow the Lyapunov framework to define a *drift-plus-penalty* function:

$$\Delta(Q(t)) + V\mathbb{E}\{D(t)|Q(t)\}, \tag{9}$$

where scalar parameter $V \geq 0$ is a constant value to control the tradeoff. By setting a larger value of V, the time-average bit-rate will become close to the optimal bound. On the other hand, with a smaller value of V, the system can keep the queue at lower backlog, which leads to lower job completion delay.

In the Lyapunov optimization, the *drift-plus-penalty* has the following upper bound [9] for all t and all $V \geq 0$,

$$\begin{aligned} F &= \Delta(Q(t)) + V\mathbb{E}\{D(t)|Q(t)\} \\ &\leq B + V\mathbb{E}\{D(t)|Q(t)\} \\ &\quad +Q(t)\mathbb{E}\{\lambda(t) - c_a(t)|Q(t)\}, \end{aligned} \tag{10}$$

where B is a positive constant that satisfies

$$\begin{aligned} B &\geq \frac{1}{2}\mathbb{E}\{\lambda(t)^2 + c_a^2(t)|Q(t)\} \\ &\quad -\mathbb{E}\{\widetilde{c}_a(t)\lambda(t)|Q(t)\}, \end{aligned} \tag{11}$$

where $\widetilde{c}_a(t) = \min[Q(t), c_a(t)]$.

For simplicity, rather than directly minimizing the drift-plus-penalty of (9), our strategy is to minimize the bound given at the right-hand-side of (10). Since $\lambda(t)$ is independent of the action $a(t)$, at each time slot we only need to minimize

$$\begin{aligned} \min_{a(t)} \quad & VD(t) - Q(t)\cdot c_a(t) \\ \text{s.t.} \quad & a(t) \in \mathcal{A}, \quad \forall t \ . \end{aligned} \tag{12}$$

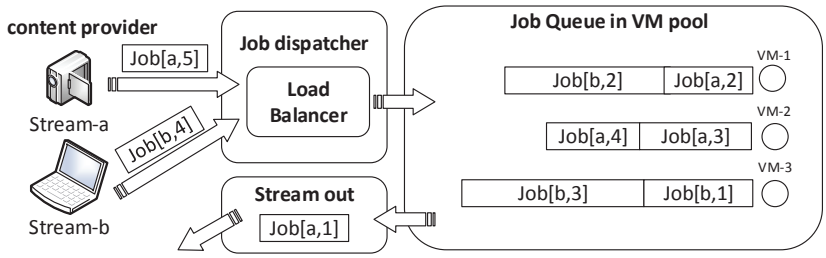


Fig. 1. The diagram of the cloud-based video transcoding system.

In (12), for each preset $a(t)$, the output bit-rate $D(t)$ and the workload execution time $c_a(t)$ can be estimated through video profiling, which is basically some statistics obtained through offline training or history data, and $Q(t)$ is known at time slot t through the regular queue update. Thus, the problem of (12) can be simply solved by sorting the values of $VD(t) - Q(t) \cdot c_a(t)$ for different $a(t)$ and choose the preset that has the smallest value. The developed procedure is summarized in Algorithm 1.

Algorithm 1 Adaptive Transcoding Configuration

procedure SCHEDULING
  for $i = 1, ..., N$ **do** // for each queue
    for $a(t) = 1, ..., k$ **do** // for each mode
      $X(a(t)) \leftarrow V \cdot D(t) - Q(t) \cdot c_{a(t)}(t)$
    end for
    use $a(t)$ where $X(a(t))$ is minimal
    update $Q(t+1)$
  end for
end procedure

IV. Simulation Results

To verify the proposed method, we choose two 10-minute 1080p full-HD videos, "big buck bunny" (BBB) and "Sintel" (STL), to do the simulation. The original video is encoded with MPEG-4 codec [10] and segmented into 5-second duration video files. Three different presets, 'superfast', 'faster' and 'medium', are considered as the configuration choices for transcoding the videos into H.264 format. All the simulations are run on a computer with 3.2GHz Xeon CPU E5-1650.

Fig. 2 shows the transcoding performance statistics under different presets. It can be seen that in general, with more complex preset, the transcoding time is increasing while the bit-rate is decreasing. However, there are cases where the'faster' preset may cost less time than the 'superfast' preset and obtain smaller bit-rate. We use normal distribution to approximate the measured statistics and assume the distribution is known. In practice, such distribution can be estimated through offline video profiling or online fitting to the historical data. With the distribution information, at each time slot we can estimate the output bit-rate $D(t)$ and the workload execution time $c_a(t)$ in (12). Note that the actual output bit-rate and the workload execution time can only be obtained after encoding the arrived video segments.

We construct a discrete event simulator, having totally 1000-second running time, to evaluate the performance. The

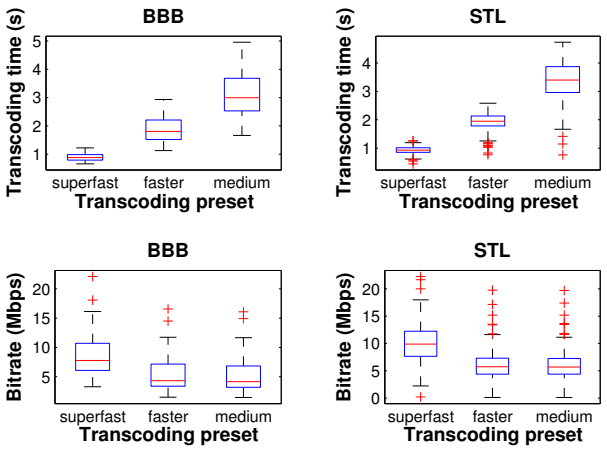
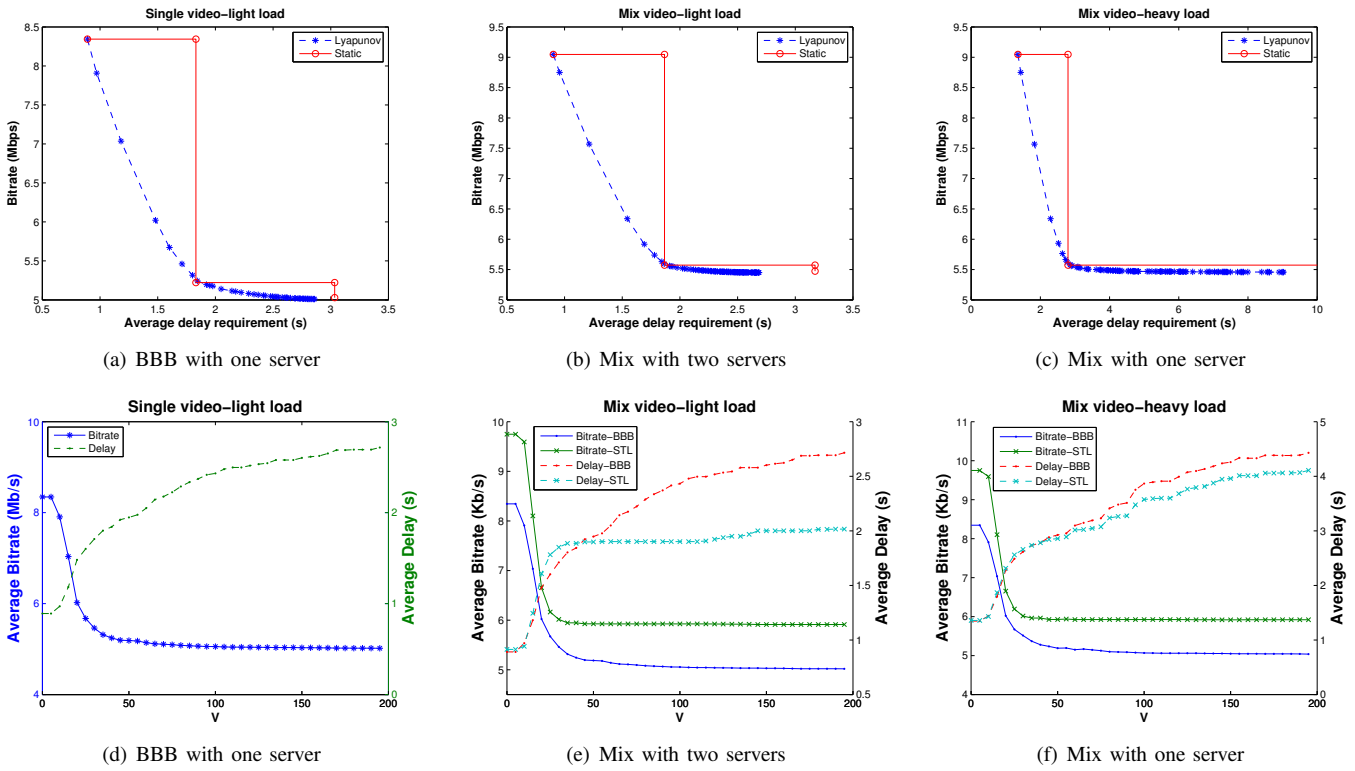


Fig. 2. Transcoding performance statistics of the workload execution time and the output bit-rate for the two real-trace videos under three presets (superfast, faster and medium). The blue box is the 95% interval, where the red line within the box is the median value. The upper and lower black lines are the bounds of the fitted normal distribution, while the red 'plus' symbols denotes the cases where the actual transcoding performance is out of the estimated range.

interval of the video segment arrival on the cloud system is set to 5 seconds, equal to the video segment duration. The video segments are repeatedly sent to the cloud system. Three scenarios are simulated: single video (BBB) with one VM or server, mix (two videos BBB and STL arrive simultaneously) with two servers, and mix with one server. In the simulation, a static configuration strategy is used as the baseline, which uses a fixed transcoding preset without aware of the video content and the queue status, to compare with our proposed adaptive configuration.

The top row of Fig. 3 shows the performance comparison with the static configuration approach, which can only fix to one of the three presets. It can be seen that, if the job delay constraint is between the average delay performance of two presets, the static method can only choose the faster one. On the contrary, the proposed method can adaptively choose the presets. At the static preset points, the proposed method achieves the same performance, while between any two static preset points the proposed method achieves much lower bit-rates than the static method. In 3(c), when system is heavily loaded, the static method with the "medium" preset suffers a very long delay of 142 seconds for a $5.47Mbps$ bit-rate. In contrast, the proposed algorithm achieves similar output bit-rate with only about 5-seconds delay.



(a) BBB with one server (b) Mix with two servers (c) Mix with one server

(d) BBB with one server (e) Mix with two servers (f) Mix with one server

Fig. 3. Simulation results. Top: Performance comparison between the static configuration and the proposed adaptive configuration. Bottom: Performance of the proposed method under different V values.

The bottom row of Fig. 3 shows the detailed trade-off curves of the proposed method. It can be seen that when the trade-off parameter V goes to more than 100, the output bit-rate is roughly the lowest. In the light load condition, we see the delay curves for the two videos reach saturation at different V. On the other hand, when the system load is heavy in the case of 'mix with one server', the delay curves for the two videos are close to each other since all jobs are equally proceeded.

V. Conclusion

In this paper, we have presented an adaptive configuration approach for cloud based video tanscoding, which exploits the tradeoff between the output bit-rate and the transcoding delay. By applying the Lyapunov framework, the arrived transcoding jobs are adaptively configured under the constraint of queue backlog stability. Simulations on real video traces show that the proposed adaptive method significantly outperforms the static configuration points at almost all the delay constrained points. Future works include experiments on real cloud test-bed and combing with long-term resource adaptation such as auto-scaling.

Acknowledgment

This research is supported by the Singapore National Research Foundation under its IDM Futures Funding Initiative and administered by the Interactive & Digital Media Programme Office, Media Development Authority.

References

[1] Z. Huang, C. Mei, L. E. Li, and T. Woo, "Cloudstream: delivering high-quality streaming videos through a cloud-based svc proxy," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 201–205.

[2] X. Wang, T. T. Kwon, Y. Choi, H. Wang, and J. Liu, "Cloud-assisted adaptive video streaming and social-aware video prefetching for mobile users," *Wireless Communications, IEEE*, vol. 20, no. 3, 2013.

[3] S. Ma, W. Gao, and Y. Lu, "Rate-distortion analysis for H. 264/AVC video coding and its application to rate control," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 12, pp. 1533–1544, 2005.

[4] W. Zhang, Y. Wen, and H.-H. Chen, "Toward Transcoding as a Service: Energy-Efficient Offloading Policy for Green Mobile Cloud," *Network, IEEE*, vol. 28, no. 6, pp. 67–73, Nov 2014.

[5] W. Zhang, Y. Wen, J. Cai, and D. 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, Jun 2014.

[6] A. Ashraf, F. Jokhio, T. Deneke, S. Lafond, I. Porres, and J. Lilius, "Stream-Based Admission Control and Scheduling for Video Transcoding in Cloud Computing," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, 2013, pp. 482–489.

[7] Y. Song, M. Zafer, and K.-W. Lee, "Optimal bidding in spot instance market," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 190–198.

[8] H. Ma, B. Seo, and R. Zimmermann, "Dynamic Scheduling on Video Transcoding for MPEG DASH in the Cloud Environment," in *MMSys, 2014 Proceedings ACM*.

[9] M. J. Neely, *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers, 2010, vol. 3, no. 1.

[10] I. E. Richardson, *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. Wiley. com, 2004.