High Performance On-Demand Video Transcoding Using Cloud Services

Xiangbo Li*, Mohsen Amini Salehi† (Co-Advisor), Magdy Bayoumi*(Advisor)
*The Center for Advanced Computer Studies
†HPCC Laboratory, Computer Science Department
University of Louisiana at Lafayette, Louisiana, USA
Email: {xxl8948, amini, mab}@cacs.louisiana.edu

***Abstract*—Video streams, either in form of on-demand streaming or live streaming, usually have to be converted (*i.e.,* transcoded) based on the characteristics (*e.g.,* spatial resolution) of clients' devices. Transcoding is a computationally expensive operation, therefore, streaming service providers currently store numerous transcoded versions of the same video to serve different types of client devices. However, recent studies show that accessing video streams have a long tail distribution. That is, there are few popular videos that are frequently accessed while the majority of them are accessed infrequently. The idea we propose in this research is to transcode the infrequently accessed videos in a on-demand (*i.e.,* lazy) manner. Due to the cost of maintaining infrastructure, streaming service providers (*e.g.,* Netflix) are commonly using cloud services. However, the challenge in utilizing cloud services for video transcoding is how to deploy cloud resources in a cost-efficient manner without any major impact on the quality of video streams. To address the challenge, in this research, we present an architecture for on-demand transcoding of video streams. The architecture provides a platform for streaming service providers to utilize cloud resources in a cost-efficient manner and with respect to the Quality of Service (QoS) requirements of video streams. In particular, the architecture includes a QoS-aware scheduling component to efficiently map video streams to cloud resources, and a cost-efficient dynamic (*i.e.,* elastic) resource provisioning policy that adapts the resource acquisition with respect to the video streaming QoS requirements.**

I. Introduction

The way people watch videos has dramatically changed over the past decades. From traditional TV systems, to video streaming on desktops, laptops, and smart phones through Internet. According to Global Internet Phenomena Report [1], video streaming currently constitutes approximately 64% of all the U.S. Internet traffic. Cisco Systems, Inc.[1] estimates that the streaming traffic will increase up to 80% of the whole Internet traffic by 2019 [2].

Video content, either in form of on-demand streaming (*e.g.,* YouTube [2] or Netflix [3]) or live-streaming (*e.g.,* Livestream [4]), needs to be converted based on the characteristics of the clients devices. That is, the original video has to be converted to a supported resolution, frame rate, video codec, and network bandwidth of the clients devices. The conversion is termed *video transcoding* [3], which is a computationally heavy and time-consuming process. Due to the limitations in processing power and energy sources (*e.g.,* in smart phones), it is not practical to transcode videos on clients' devices. In addition, provisioning and maintaining in-house infrastructures to meet the fast-growing demands of video transcoding is cost-prohibitive. Therefore, streaming service providers (*e.g.,* Netflix) have become reliant on cloud services.

One approach currently used to support a diversity of client devices is to transcode and store numerous versions of the same video in advance (called pre-transcoding). However, pre-transcoding requires massive storage and processing capabilities. Given the explosive growth of the video streaming demands on a large diversity of the client devices, this approach remains unachievable, specifically for small- and medium-size streaming service providers. Interestingly, recent studies has shown that accessing video streams exhibits a long tail distribution [4]. That is, just few popular videos are accessed frequently and the vast majority of them are rarely accessed. We argue that there is no need to pre-transcode videos for infrequently accessed videos. Thus, our proposal in this research is to transcode video streams in an on-demand (*i.e.,* lazy) manner using computing services offered by cloud providers. Then, the challenge of this research, is how to employ cloud services in a cost-efficient manner and without a major impact on the QoS demands of video streams.

Video stream clients have unique QoS demands. In particular, they need to receive video streams without any delay. Such delay may occur either during streaming, due to an incomplete transcoding task by its presentation time, or it may occur at the beginning of a video stream. In this research, we refer to the former delay as *missing presentation deadline* and the latter as the *startup delay* for a video stream. Previous studies (*e.g.,* [5]) confirm that streaming clients mostly do not watch videos to the end. However, they rank the quality of a stream provider based on the video's startup delay. Therefore, to maximize clients' satisfaction, we formally define video streaming QoS demand as: minimizing the startup delay without missing the presentation deadline.

Streaming service provider's goal is to spend the minimum for cloud resources, while meets the QoS requirements of video streams. Satisfying this goal becomes further complicated when we consider the variations exist in the demand rate of video streams and different type of services offered by cloud providers (*i.e.,* heterogeneous cloud servers). To minimize the cost of utilizing cloud resources, our system should adapt its service rate (*i.e.,* transcoding rate) based on the clients' request rate while respecting the video streams QoS requirements.

Based on the provided definitions, the main research questions we address in this research are:

[1]http://www.cisco.com/
[2]https://www.youtube.com
[3]https://www.netflix.com
[4]https://livestreams.com

- How to perform on-demand video transcoding while providing required QoS for video on-demand (VOD) and live streaming?
- How to minimize the incurred cost of streaming service providers to perform on-demand video transcoding, while the clients' QoS demands are respected?

To answer these challenges, in this research, we propose an architecture that enables a stream provider to utilize cloud resources with the minimum incurred cost and maximum client satisfaction. The architecture is able to support both live and VOD streaming requests at the same time. In summary, the key **contributions** of this research are:

- Proposing a QoS-aware video transcoding using homogeneous cloud services, discussed in section III.
- Proposing a cost-efficient video transcoding using heterogeneous cloud servers, presented in section IV.
- Proposing a high performance cloud-based hybrid (*i.e.,* live streaming and VOD) video transcoding, described in section II and V.
- Developing a prototype of online hybrid video transcoding using cloud services, presented in section V.

In previous works (*e.g.,* [6], [7]), cloud-based video transcoding methods were proposed for video on-demand (VOD). However, their focus are mainly on efficiently utilizing cloud resources for offline (not on-demand) transcoding. They do not consider clients' QoS requirements. As for live video streaming, recent studies [8], [9] focus on how to transfer live streaming content, while live streaming transcoding has remained intact.

Our research is different from Content Delivery Networks (CDN) where data (*e.g.,* video contents) are distributed in various geographical areas for quick access. In fact, our on-line transcoding system can complement CDNs by customizing the video formats based on the viewers' devices characteristics.

II. Hybrid Video Transcoding (HVT) Architecture

We present the HVT architecture for transcoding a mixture of video on-demand (VOD) and live streaming using cloud services. An overview of the architecture is presented in Figure 1. The architecture shows the sequence of actions taken place when clients (*i.e.,* viewers) request videos from a streaming service provider. The architecture includes seven main components, namely *video splitter, transcoding time estimator, task (i.e., GOP) scheduler, transcoding virtual machines (VM), elasticity manager, video merger,* and *caching policy*. The cooperation of these components leads to cost-efficient and QoS-aware video transcoding on the cloud. These components are explained in the next few subsections.

Video Splitter: A Video stream consists of several sequences. Each sequence is divided into multiple independent Group Of Pictures (GOP). In this work, we treat each GOP as a task with an individual presentation deadline. In VOD streaming (*e.g.,* Netflix and YouTube), if a GOP misses its deadline, it still has to complete its transcoding whereas in

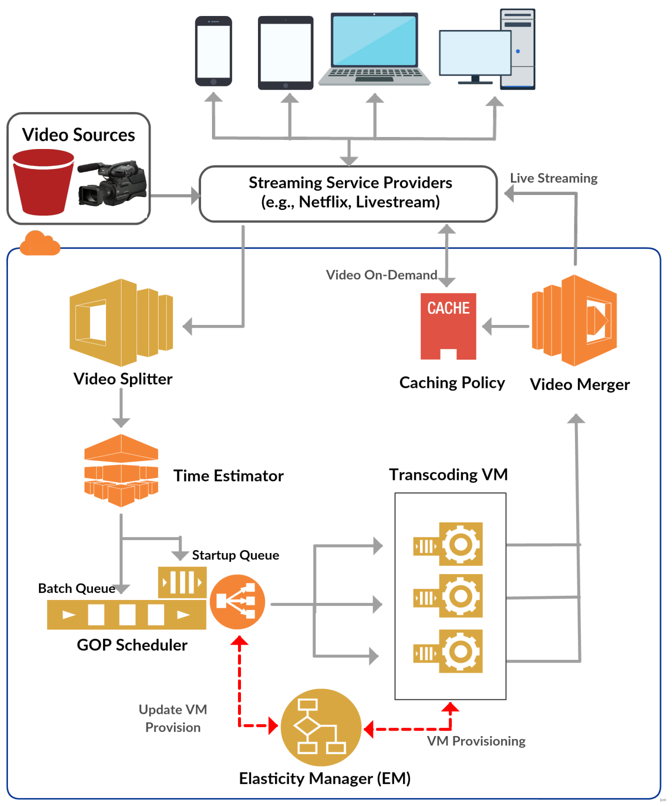

Fig. 1: An overview of the Hybrid Video Transcoding (HVT) architecture

live streaming, a GOP is dropped (discarded) if it misses the deadline.

Transcoding Time Estimator: In VOD streaming, a video usually is streamed multiple times by different clients. Therefore, an estimation of the transcoding execution time for each GOP, can be obtained from the historic execution information of that GOP. However, there is no such historic execution time information is available in live streaming.

Transcoding (GOP) Task Scheduler: It is responsible for mapping GOPs to transcoding servers. The scheduler's goal is to satisfy the QoS demands of the clients in terms of minimum startup delay and minimum deadline violation.

Transcoding Virtual Machine (VM): VM(s) are allocated from the cloud provider to transcode GOP tasks. Each VM has a local queue where the required data for GOPs are preloaded before execution. Whenever a free spot appears in the local queue of a VM, the scheduler is notified to map a GOP to the VM. We assume that the GOP tasks in the local queue are scheduled in the FCFS fashion.

Elasticity Manager (EM): EM monitors the operation of transcoding VMs and accordingly resizes the VM cluster with the goal of meeting the clients QoS demands and minimizing the incurred cost to the stream provider.

Video Merger: It places all the transcoded GOPs in the right order and creates the resulting (*i.e.,* transcoded) video stream.

Caching Policy: To avoid unnecessary transcoding of the trending videos, the HVT architecture provides a caching policy to decide whether a transcoded video should be cached or not. This component is not used for live streaming.

III. QoS-aware video transcoding Using Homogeneous Cloud Services

In this section, we investigate the feasibility of on-demand video transcoding using cloud services without major QoS violation for the video streams. We consider VOD type of streaming where each transcoding task (*i.e.,* GOP) has an individual deadline and it has to complete even if it misses its deadline. In addition, historic execution times of the transcoding tasks are available. We propose a scheduling method whose goal is to minimize the startup delay for the new streams while meeting the deadline of other GOPs. Moreover, we develop a provisioning policy for the Elasticity Manager component of the architecture. The provisioning policy monitors the operation of cloud VMs and dynamically resizes (*i.e.,* allocates/deallocates) VMs to minimizes the incurred cost to the provider without violating their QoS requirements.

A. QoS-Aware Scheduling Method

For scheduling, GOPs of the requested video streams are batched in a queue upon arrival. To minimize the startup delay of video streams, we considered another queue termed *startup queue*. The first few GOPs of each new video stream are placed in the startup queue that has a higher priority in compare to the batch queue. For each GOP j from video stream i, denoted G_{ij}, the arrival time and the deadline (denoted δ_{ij}) are available.

Although GOP tasks in the startup queue have a higher priority, they should not cause deadline violation for tasks waiting in the batch queue. Let G_b, the first GOP in the batch queue and G_s, the first GOP in the startup queue. At each scheduling event, G_s can be scheduled before G_b only if it does not cause G_b to miss its deadline. For that purpose, we calculate the minimum completion time of G_s across all VMs. Then, we can calculate the minimum completion time of G_b, assuming that G_s has already been mapped to a VM, and finally check if G_b will miss its deadline or not. If not, then G_s can be scheduled before G_b. More details about the scheduling policy can be found in our previous publication [10].

Figure 2 demonstrates that using the QoS-aware scheduling, we can keep the average startup delay less than 1 second. The startup delay remains almost the same as the number of video streams increases. More importantly, results also show that the reduced startup delay is obtained without a major impact on the video streams' deadline miss rate.

B. Dynamic Resource Provisioning Policy

We combine both resource utilization and QoS violation factors in the resource provisioning policy. The provision policy occurs periodically at *provisioning events* to make allocation or deallocation decisions. At each provisioning event, the policy predicts the deadline miss rate that will occur at the next provisioning event based on the current state of the local queues and the batch queue. We also proposed a lightweight remedial resource provisioning policy that can improve the efficiency of the EM component in the architecture. By injecting this policy to the intervals of the periodic provisioning

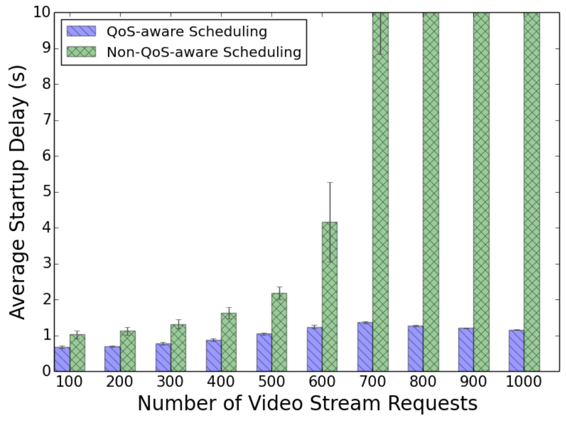

Fig. 2: Startup delay of various videos [10].

policy, we can perform the periodic policy less frequently. The remedial provisioning policy provides a quick prediction of the system based on the state of the startup queue. The algorithm and more details are discussed in our previous work [10].

The provisioning policy tries to keep the QoS violation (*i.e.,* deadline violation) within a certain thresholds of α and β. Therefore, the performance of the provisioning policy is dependent on these thresholds. Figure 3 shows the study we have accomplished based on the Pareto optimal front based on different upper bound threshold β that the stream provider can choose. As we can see, the lower β value produces lower startup delay and deadline miss rate, but also incurs higher cost. In contrast, higher β value reduces the expense at the cost of higher QoS violation. However, for β values between 0.15 to 0.3 our provisioning policy provides a relatively low QoS violation and reasonably low cost. We noticed that the relationship between the cost and QoS violation in our system is not linear. That is, there are some optimal points where a stream provider can spend a relatively low cost and gain a fairly low QoS violation too.

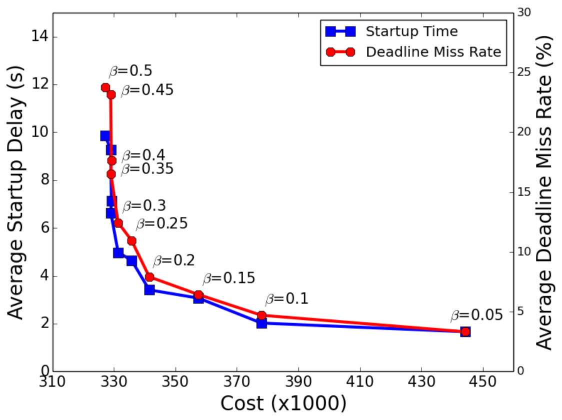

Fig. 3: Illustration of the Pareto front for determining the upper bound threshold (β) in the dynamic provisioning policy [10].

IV. Cost-efficient video transcoding using heterogeneous cloud servers

In evaluating the feasibility of on-demand transcoding, we realized that there is an affinity between the type of transcoding task and the VMs offered by the cloud providers. For instance, some transcoding tasks (*e.g.,* changing codec) are compute-intensive and has a high affinity with GPU-based VMs whereas other transcoding tasks (*e.g.,* changing resolution) are memory-intensive and a VM with a large memory is a better match for them. Taking into account the affinity between the transcoding tasks and VM types can potentially lead to solutions that are more cost-efficient and provider a lower QoS violation. Therefore, as the next step in this research, we are currently researching on new scheduling methods and resource provisioning policies that utilize heterogeneous VMs offered by the cloud providers.

In our previous work [10], [11], the scheduling method and the resource provisioning policy are based on homogeneous VMs. That is, each GOP has the same transcoding time on different VMs. However, the scheduling method and the resource allocation policy cannot be applied in a heterogeneous system, because GOP transcoding time is varies on different VM instance types. In addition, in this part of the research, we extend the concept of startup delay to all GOPs of an stream by providing the concept of utility function for each GOP task. That is, to prioritize GOPs in the beginning of the stream, we offer them a higher utility value and this value diminishes as we move to the later GOPs in the stream. Then, the goal of the scheduling method is to maximize the utility earned by the GOPs.

At each scheduling event, the scheduler constructs a virtual queue based on the GOPs in the batch queue. The virtual queue includes the GOPs with the highest utility value from each video stream request in the batch queue. Then, our proposed scheduling method will be applied on the GOPs in the virtual queue with the goal of maximizing the utility value and without violating the deadline of other GOPs.

Regarding heterogeneous resource provisioning policy, the question is not limited to *when* and *how many* VMs should be allocated or deallocated. In fact, it includes one more dimension and that is *which type* of VM instance should be allocated or deallocated. Previous studies [12] show that a few less powerful instances perform better than one powerful VM and it is cheaper. To provide high QoS demands with minimum incurred cost, our new proposed resource allocation policy will be based on three main factors, namely *QoS violation, resource utilization rate,* and *cost of VM instance*.

V. High Performance Cloud-based Hybrid Video Transcoding

The Hybrid Video Transcoding (HVT) architecture has been discussed in Section II. In this architecture, the system deals with a combination of live streaming and VOD transcoding tasks. Supporting live video streaming and VOD at the same time in a system introduces new challenges.

Live streaming transcoding tasks are different from VOD streaming from several aspects. Most importantly, there is no historic data of the GOP execution times to estimate the execution time of the current GOP. And, unlike VOD, the GOP tasks need to be dropped once they miss their deadlines. These differences will make the scheduling of the tasks in hybrid environment a more complicated problem.

The last phase of my research will be developing a prototype of online hybrid video transcoding using cloud services.

VI. Conclusion

We proposed the HVT architecture for on-demand transcoding of video streams using cloud resources in our prvious work [10]. Experiment results show that our proposed scheduling method provides low QoS violation rate. In addition, the dynamic resource provisioning policy helps streaming providers to significantly reduce the cost of using cloud services. In particular, when the video demand rate is not high, it reduces the costs up to 70% in compare with the static policies. An improved cost-efficient and QoS-aware architecture based on the heterogeneous VMs will be discussed in chapter two of the thesis. At the end of our work, we will proposing and developing a prototype of online hybrid video transcoding using cloud services. The goal of this work is to decrease QoS violation of video streams and minimize the incurred cost of cloud resources for streaming service providers. The proposed architecture can be particularly useful for small- or medium-size video streaming provides to utilize cloud services as their infrastructure, and improve their clients' satisfaction with low cost.

References

[1] G. I. P. Report, "https://www.sandvine.com/trends/global-internet-phenomena/," accessed Oct. 1, 2015.

[2] C. V. N. Index, "Forecast and methodology, 2014-2019," 2015.

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

[4] N. Sharma, D. K. Krishnappa, D. Irwin, M. Zink, and P. Shenoy, "GreenCache: augmenting off-the-grid cellular towers with multimedia caches," in *Proceedings of the 4th ACM Multimedia Systems Conference*, pp. 271–280, 2013.

[5] X. Cheng, J. Liu, and C. Dale, "Understanding the characteristics of internet short video sharing: A youtube-based measurement study," *IEEE Transactions on Multimedia*, vol. 15, no. 5, pp. 1184–1194, 2013.

[6] F. Jokhio, A. Ashraf, S. Lafond, I. Porres, and J. Lilius, "Prediction-based dynamic resource allocation for video transcoding in cloud computing," in *Proceedings of the 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 254–261, 2013.

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

[8] A. H. Payberah, H. Kavalionak, V. Kumaresan, A. Montresor, and S. Haridi, "Clive: Cloud-assisted p2p live streaming," in *Proceedings of the 12th IEEE International Conference on Peer-to-Peer Computing (P2P)*, pp. 79–90, 2012.

[9] F. Wang, J. Liu, and M. Chen, "CALMS: Cloud-assisted live media streaming for globalized demands with time/region diversities," in *Proceedings of the IEEE INFOCOM*, pp. 199–207, 2012.

[10] X. Li, M. A. Salehi, M. Bayoumi, and R. Buyya, "CVSS: A Cost-Efficient and QoS-Aware Video Streaming Using Cloud Services," in *Proceedings of the 16th ACM/IEEE International Conference on Cluster Cloud and Grid Computing (CCGrid '16)*, May, 2016.

[11] X. Li, M. A. Salehi, and M. Bayoumi, "Cloud-Based Video Streaming for Energy- and Compute-Limited Thin Clients," in *the Stream2015 Workshop at Indiana University*, Oct, 2015.

[12] B. Javadi, R. K. Thulasiram, and R. Buyya, "Statistical modeling of spot instance prices in public cloud environments," in *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC)*, pp. 219–228, 2011.