HLSAAS: HIGH-LEVEL LIVE VIDEO STREAMING AS A SERVICE

Mohsen Amini Salehi[1], *Xiangbo Li*[2]

[1]HPCC Laboratory, Computer Science Department
[2]The Center for Advanced Computer Studies
University of Louisiana Lafayette, LA 70504, USA
{amini,xxl894}@cacs.louisiana.edu

1. INTRODUCTION

An increasingly popular type of streaming service is *live video streaming* that enables users to broadcast videos via the Internet while they are captured. For instance, using Livestream[1], viewers are able to watch the contents being captured by users on their smart phones, laptops, and TVs.

Thanks to the high speed Internet, basic live video streaming is feasible nowadays. However, what is offered currently is far from the high-level services and qualities demanded by the viewers and video providers. *High-level streaming services* is defined as any types of further processing on the streaming videos. For instance, the current basic live streaming providers support a single generic display format and the viewers have to adapt their devices (*e.g.,* in terms of compression standards) to this supported format. However, viewers' demand is to have a high quality video stream converted (*i.e.,* transcoded) based on the characteristics of their display devices. Another instance of high-level video streaming services, is to preserve the privacy of captured objects (*e.g.,* people faces or car plate numbers) in the streaming videos. These instances will be described in more details in Section 2.

The challenge in further processing of the video streams is twofold: *first,* video processing is computationally expensive and requires huge computing infrastructures; *second,* the video processing has to be carried out in a real-time manner to fulfill the QoS demands of live streaming viewers.

To address these challenges, streaming providers are increasingly becoming reliant on cloud services for their computational needs. The problem in utilizing cloud services, however, is to spend the minimum cost for the services while meeting the viewers' QoS demands. In particular, live streaming viewers have unique QoS demands. They need to receive video streams without any delay. That is, the video processing tasks should be completed within a short deadline (*i.e.,* before their presentation times). Tasks that miss their presentation times must be dropped (*i.e.,* discarded) to keep up with the live streaming. Accordingly, the *drop rate* is defined as the percentage of tasks that cannot complete before their presentation deadlines. In addition, viewers generally judge the streaming providers based on the delay in the first few seconds of the stream, termed the *startup delay*. Under this circumstance, to maximize viewers' satisfaction, video streaming QoS demand is defined as minimizing the startup delay and the drop rate.

To support a flexible range of high-level live video streaming services, we present an architecture called High-level Live Streaming as a Service (HLSaaS). The architecture is able to accept any high-level video processing request and apply that on the live video streams. According to the requested high-level service, the HLSaaS architecture allocates computational resource from cloud to minimize the incurred cost while respecting the QoS demands of the viewers.

2. HIGH-LEVEL LIVE VIDEO STREAMING

A Video stream consists of several sequences. Each sequence is further divided into multiple *Group Of Pictures* (GOP) with sequence header information in the beginning. GOP is essentially a sequence of frames beginning with an I (intra) frame, followed by a number of P (predicted) frames or B (be-directional predicted) frames. We consider each GOP as the processing unit for video stream processing.

There are various high-level live streaming services that can take advantage of our proposed HLSaaS architecture. For each of these applications, the architecture fulfills the streaming QoS demands and minimizes the incurred cost to the stream provider. Below, we explain two of such applications.

2.1. Privacy-aware Live Video Streaming

In live video streaming, while the video is being captured, it is streamed and presented to the users in a real-time manner. However, usually there are unintended contents in the videos that can reveal unauthorized information and threats the privacy. For instance, commonly, people's faces are unintentionally captured while we record and stream a video from a scene. Another example is the car plate numbers that are captured unintentionally when we stream a video. How-

[1]https://livestream.com/

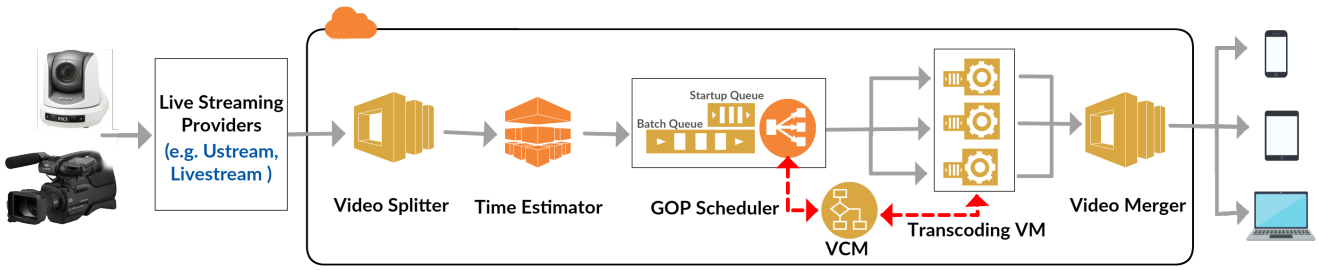

Figure 1: An overview of HLSaaS: A High-level Live Streaming as a Service architecture

ever, people or car owners may not like to be recognizable in the video streams.

Currently, there is no solution to blur or remove the unauthorized contents from the frames in a live video stream. The HLSaaS architecture enables video broadcasters to blur the unauthorized parts before their contents are reached to the viewers while it maintains the cost-efficiency and streaming QoS demands.

2.2. Live Video Transcoding

Captured videos usually should be converted (*i.e.,* transcoded) from the original format based on the characteristics of the viewers devices (*e.g.,* supported spatial resolution, frame rate, and bit rate).

The HLSaaS architecture enables stream providers to support a wide range of viewers' display devices by introducing transcoding operations for those devices.

3. HLSAAS ARCHITECTURE

We propose an architecture to provide high-level live video streaming services using cloud. The architecture (presented in Figure 1) shows the sequence of actions taken place when viewers request videos from a live streaming service provider. Cooperation of these components leads to cost-efficient and QoS-aware live streaming. These components are as follows:

Video Splitter: Splits each video into several GOPs, that can be processed independently. Each GOP has an individual deadline based on the presentation time of the first frame in that GOP. In live streaming, if a GOP misses its deadline, it is dropped. **Execution Time Estimator:** In live streaming, the execution time of GOP tasks are unknown. This component estimates the execution time using machine learning techniques.

(GOP) Task Scheduler: is responsible for mapping GOPs to the processing servers (VMs). The scheduler goal is to satisfy the QoS demands of clients in terms of minimum startup delay and GOP drop rate of video streams.

For scheduling, GOPs of the requested video streams are batched in a queue upon arrival. To minimize the startup delay, we consider another queue termed the *startup queue* with higher priority. The first few GOPs of each new video stream are placed in the startup queue. The scheduler maps GOPs to the VM that provides the shortest completion time. We assign a higher priority to the GOPs of the startup queue. However, the priority should not cause missing the deadlines of tasks waiting in the batch queue.

Stream Processing Virtual Machines (VM): VM(s) are allocated from the cloud provider to process GOP tasks. Each VM has a local queue where the required data for GOPs are preloaded before execution. We assume that the GOP tasks in the local queue are scheduled using the FCFS method. **Virtual Cluster Manager (VCM):** Monitors the operation of VMs, and resizes the VM cluster to meet the viewers' QoS demands and to minimize the incurred cost of the streaming provider. **Video Merger:** places all the processed GOPs in the right order, creates the live stream, and sends the processed live streams to the viewers.

4. PERFORMANCE EVALUATION

We evaluated the scheduling of the HLSaaS for live video transcoding service. For that purpose, we varied the number of live streaming requests and measured the average startup delay of the streaming videos. Figure 2 demonstrates how the average startup delay of live streams is reduced when the proposed QoS-aware scheduling method is applied in compare with a non QoS-aware scheduling method. We observe that by using the QoS-aware scheduling, the average startup delay is less than 1 second. Plus, the startup delay remains almost constant as the number of live streaming requests increases.

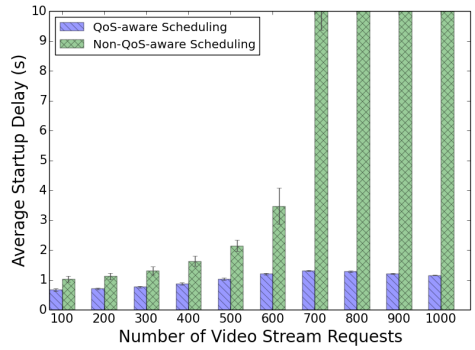

Figure 2: Average startup delay for various number of live streaming video requests.