Cost-Efficient Video On Demand (VOD) Streaming Using Cloud Services

A Dissertation

Presented to the

Graduate Faculty of the

University of Louisiana at Lafayette

In Partial Fulfillment of the

Requirements for the Degree

Doctor of Philosophy

Mahmoud K. Darwich

Fall 2017

Cost-Efficient Video On Demand (VOD) Streaming Using Cloud Services

Mahmoud K. Darwich

APPROVED:

Mohsen Amini Salehi, Co-Chair Assistant Professor of Computer Science The Center for Advanced Computer Studies	Magdy Bayoumi, Co-Chair Professor of Computer Engineering The Center for Advanced Computer Studies
Chee-Hung Henry Chu Professor of Computer Engineering The Center for Advanced Computer Studies	Ashok Kumar Associate Professor of Computer Science The Center for Advanced Computer Studies

Mary Farmer-Kaiser
Dean of the Graduate School

DEDICATION

To the loves of my life

Reem Salhab, Hussam, and Kareem

ACKNOWLEDGMENTS

All praise is due to the almighty God ALLAH, the Most Gracious and the Most Merciful for his blessings bestowed upon me (as well as mankind) and for giving me the strength to achieve what I have accomplished in my life. This work is a collective result of the efforts, advice, and opinions of many people.

First and foremost, I would like to express my extreme gratitude to my advisors Dr. Mohsen Amini Salehi and Dr. Magdy Bayoumi for their guidance, help, support, and patience during my stay with the Center for Advanced Computer Studies (CACS). Their guidance helped me during the research time and writing of this thesis. I could not have imagined having better advisors and mentors for my PhD study. My respect and thanks also go to the members of my committee: Dr. Chee-Hung Henry Chu and Dr. Ashok Kumar for their tremendous advice, insightful comments, and encouragement.

My family played an important role both in my life and my study and it is an honor to dedicate this work to them as their support and encouragement were in the end what made this work valuable. I would like to thank my parents Khaled and Khadija for their prayers, love, and faith in me and the many years of support and encouragement throughout my whole life. My deepest appreciation goes to my lovely wife Reem and my precious sons Hussam and Kareem. My wife always makes me feel that I am the luckiest man. She believed in me and she stood by my side when I most wanted her, and helped me to be the best I can be through her patience, support, and wonderful love. Her support and encouragement were in the end, what made this dissertation possible.

Last but not least, I would like to thank my brothers and friends, and I will start with my best ever brothers Dr. Talal Darwich, Dr. Tarek Darwish, Taha Darwich, and

Mohamad Darwich for their prayers, encouragement, and support. I would also to thank warmly Ege Beyazit, Dr. Xiangbo Li, Dr. Ahmed Abdelgawad, and Dr. Abdelhamid Morsy for their useful discussions, support, and help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: Introduction

Video streaming is one of the increasingly popular services in the computing and networking industry. A wide range of applications is reliant on video streaming. The examples of such systems are e-learning systems [1], video chat and conferencing systems [2], natural disaster management, and security systems that operate based on video surveillance [3], network-based broadcasting channels (*e.g.,* news and other TV channels) [4], personal broadcasting through social networks, and the movie industry.

Receiving a stream of video content is not a new idea, and it dates back to the invention of television in the early years of the 20th century. However, the medium through which people receive and watch video content has dramatically changed over the past years, from traditional TV systems to streaming on desktops, laptops, and smartphones through the Internet. Consumer adoption of the new video streaming services is skyrocketing. Based on the Global Internet Phenomena Report [5], video streaming currently constitutes approximately 64% of all the U.S. Internet traffic. It is estimated that streaming traffic will constitute up to 80% of all Internet traffic in coming years. With different architectural characteristics, viewers stream videos on a variety of devices, from large screen TVs and desktops to tablets and smartphones. Video contents, either in the form of Video On Demand (VOD) (*e.g.,* YouTube[a] or Netflix[b]) or live-streaming (*e.g.,* Livestream[c]), need to be converted (*i.e.,* transcoded) based on the characteristics of viewers' devices (*e.g.,* in terms of frame rate, resolution, and network bandwidth) [6].

To make the video streams readily available for viewers, video stream providers commonly carry out the transcoding operation in an offline manner. That is, they

[a] https://www.youtube.com
[b] https://www.netflix.com
[c] https://livestreams.com

transcode and store (*i.e., pre-transcode*) multiple formats of the same video to satisfy the requirements of viewers with heterogeneous display devices. In practice, video stream providers such as Netflix have to pre-transcode 40 to 50 formats of a single video [7] and store them in their repositories.

To overcome the storage and computational demand of transcoding, Video Stream Providers extensively utilize cloud computing services [8]. Pre-transcoding videos imposes a significant overhead cost to video stream providers [9], [10]. Video stream providers are seeking solutions to reduce their overhead incurred cost of using clouds.

1.1. Motivations

Recent studies show that accessing videos of a video stream provider follows a long-tail distribution [11]. That is, there are few videos that are accessed very frequently (*i.e., hot* videos) while there is a huge portion of videos in the repository that are rarely accessed. Thus, research has been undertaken (*e.g.,* [10]) to alleviate the cost of pre-transcoding by transcoding rarely-accessed videos in an on-demand (*i.e.,* lazy) manner. In this manner, one or few formats of a video is stored, and transcoding is performed on-the-fly upon request to access a version of the video that is not already pre-transcoded. This has become feasible with the enormous computational capacity clouds offer and is called *re-transcoding* video streams.

Re-transcoding induces the computational cost of video stream providers which is generally more expensive than storage cost [12]. The computational cost is so high because computation power is generally more expensive than storage services in clouds. Therefore, the re-transcoding approach would be beneficial to video stream providers, only if it is applied on the rarely accessed videos. Conversely, if it is applied on *frequently*

accessed videos (commonly known as hot videos), it increases the cost overhead even more significantly as the video stream providers have to pay for every time the video stream is transcoded. Therefore, to reduce the incurred cost to video stream providers, they have to apply pre-transcoding on frequently accessed videos and apply re-transcoding on the rest of video streams.

1.2. Research Problem and Objectives

The question arises is *what is a hot video?* and *how can we quantify the hotness of a video in a repository?* Once we answer the foregoing question, we can decide to pre-transcode hot videos and re-transcode non-hot videos. We also know that the view rate for all parts of a video stream is not the same [13]. In fact, previous studies show that even within a video stream, there is a long-tail access pattern. That is, the beginning part of video streams are accessed more often than the rest of them. Then, to further reduce the cost, we propose to *partially-transcode* videos whose hotness measure is between hot and non-hot quantities. However, the challenge is how to achieve partial-transcoding? How do we know which parts of a video should be re-transcoded and which parts pre-transcoded?

To address these challenges, in this dissertation, we propose a method to quantify the hotness of a video. For videos that are not hot, *i.e.,* do not belong to the set of hot videos, we propose a method to perform pre-transcoding on a portion of the video.

1.2.1. Objectives

Video stream providers, like YouTube, Netflix, *etc.*, store an enormous number of video streams in their repositories. Moreover, for each video stream, they keep many versions of it in order to satisfy the specifications of all users' devices. Thus, this complicates storage

and processing all video streams and becomes costly when relying on classical servers. Therefore, cloud services emerged as an alternative technology which became trustworthy by video stream providers to process the videos. However, processing video streams in the cloud still incur high cost relatively. In this dissertation, the objective is to come up with methods to manage video streams in repositories. Such methods reduce the incurred cost of using cloud services paid by video stream providers.

1.3. Contributions

In summary, the core contributions of this dissertation are as follows:

- Providing a comprehensive survey on potentials and challenges of employing clouds for video streaming.
- Analyzing the execution time of the transcoding operation on heterogeneous cloud VMs and finding out their relationship with the video content type. Also, understanding influential factors to predict execution time of video streaming tasks.
- Providing a model to identify the degree of suitability of each VM type for a given GOP.
- Providing a formal definition for hot videos and a method to quantify the hotness of video streams in the repository of a video stream provider. Also proposing methods to reduce the incurred cost of using cloud services through pre-transcode, re-transcode, or partially pre-transcode videos in the repository under long-tail and non-long-tail access patterns to videos.
- Proposing video streams clustering in a repository such that proposed partial pre-transcoding methods take minimum time when running on video streams in a

repository.

1.4. Methodology

We provide a background on video streaming and address the ways to stream videos to users. We also discuss the challenges when processing video streams and the delivering methods as well as addressing the storage and security issues. Afterwards, cloud-based video streaming is introduced as a solution for video stream providers, because cloud services offer high reliability and scalability of resources. Such resources ease the processing of video streams on VMs cloud to users with relatively less cost. Accordingly, we explain generally how the transcoding operations and packing are done in the cloud, in addition to the means of delivering video streams to users as well as the storage of video streams in the cloud.

We then perform an analysis of cloud-based video transcoding. First, we conduct analysis on the heterogeneity of VMs cloud in terms of performance, storage, and costs. In particular, we analyze the operations of video transcoding on different types of VMs. An analysis is conducted for different types of video transcoding operations on different VMs. It is crucial to have such analysis to observe the impact of GOP size in video streams on transcoding time. The analysis results in an important correlation between transcoding time and GOP size. This correlation is an essential clue in the next chapter to synthesize video streams repositories.

The trade-off cost between transcoding and storing video streams in the cloud is one of the challenges faced by video stream providers. Therefore, we propose methods to decide when a video stream should be stored or transcoded. For the sake of accuracy and precise results, we need a massive number of video streams. Thus, based on the

correlation obtained from the analysis of video transcoding previously conducted, we synthesize repositories of video streams to evaluate our proposed methods. The synthesis of video streams has based on the characteristics of real transcoded videos. Each repository contains 100,0000 synthesized video streams. Based on the number of views, we model the distribution of video streams in repositories and which follows a long-tail distribution. We further model the access pattern to the video stream. Then, we apply the proposed methods on the synthesized videos streams to manage the repositories in the cloud. We perform analysis on different scenarios of video streams repositories. The simulation results show, when applying our proposed methods to manage video streams repositories, the incurred cost of using cloud resources is reduced by up to 72%.

1.5. Dissertation Structure

The core chapters of this dissertation derive from research papers published or submitted during the course of the PhD candidature. The dissertation chapters are as follows:

- Chapter 2 provides a background on video streaming. It addresses challenges and issues in video streaming. In particular, this chapter discusses the types of video streaming, transcoding, and transmuxing. Moreover, it addresses the security that is involved when streaming a video, in addition to the techniques to store video streams and the ways to deliver them to overcome the delay. The second part of this chapter talks about video streaming in the cloud. More specifically, it discusses video transcoding and transmuxing on the cloud as well as the delivery method for video streams.

  - X. Li, M. Salehi, **M. K. Darwich**, J. Woodworth, M. Bayoumi. Cloud-Based

Video Streaming Services: A Survey. *Ready for submission to ACM Computing Surveys, 2017*

- Chapter 3 provides a background on the computational complexity of the transcoding process and the performance of it on different types of computational resources (Virtual Machines) offered by clouds. Then it presents the analysis of execution time of different video transcoding operations on various VM types in the cloud. In this chapter, the affinity of video transcoding operations with heterogeneous VMs are analyzed. Moreover, the impact of factors such as video content type, GOP size, and frames on transcoding time is analyzed. On the other hand, the trade-off between performance and cost of transcoding tasks is modeled based on heterogeneous cloud virtual machines (VMs).

  - X. Li, Y. Joshi, **M. K. Darwich**, B. Landreneau, M. A. Salehi, and M. Bayoumi. Performance Analysis and Modeling of Video Transcoding Using Heterogeneous Cloud Services. *Submitted to IEEE Transactions on Parallel and Distributed Systems (TPDS), 2017*

- Chapter 4 addresses the problem of video streams repository management. In this chapter, we model the system and we quantify the hotness of video streams. After that, the partial pre-transcoding methods are proposed. Then, the proposed methods are evaluated and compared to previous methods in the literature.

  - **M. Darwich**, E. Beyazit, M. A. Salehi, M. Bayoumi. Cost Efficient Repository Management for Cloud-Based On-demand Video Streaming. *Proceedings of the 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), San Francisco, April 2017.*

- **M. Darwich**, M. A. Salehi, E. Beyazit, M. Bayoumi. Cost Efficient Cloud-Based Video Streaming Based on Quantifying Video Stream Hotness. *Submitted to The Computer Journal, 2017.*

- Chapter 5 shows the clustering method of video streams in the repository to reduce the execution time of proposed partial pre-transcoding methods when applying on video streams repositories.

- Chapter 6 concludes this dissertation and shows the direction of future works of this research.

CHAPTER 2: Background, Taxonomy, and Context

2.1. Introduction

To achieve video streaming, either in the form of Video On Demand (VOD) (*e.g.,* YouTube[a] or Netflix[b]) or live-streaming (*e.g.,* Livestream[c]), the video content generated by a camera has to be processed in several steps before being delivered on viewers' display devices. The video contents originally generated by cameras are voluminous and cannot be transferred over the network. Therefore, in the first place, the generated video stream has to be compressed. However, there are different encoding (*i.e.,* compression) methods, and the encoded video has to be converted (also known as transcoded) based on the encoding supported by the viewers' display device. Once the video stream is transcoded, it has to be packaged in the format recognized and distribution protocol supported by the viewer's device. In practice, and particularly to have a high-level streaming quality, video contents need to be further transcoded based on the characteristics of the viewers' devices, such as frame rate, video codec, and network bandwidth, to match the viewers' devices [6].

Once the video is processed based on the viewer's display device, a distribution network is used to deliver the video stream content to viewers. Although the delivery, in theory, can be considered as a regular client-server connection over the network, in practice, it is achieved through large distributed systems known as Content Delivery Networks (CDN) [14]. Other delivery methods, such as Peer-to-peer [15] also exist for effective distribution of video streams.

To provide high-quality video streaming with a minimum delay for such diverse viewers, Streaming Service Providers (SSPs) commonly pre-process (also known as

[a] https://www.youtube.com
[b] https://www.netflix.com
[c] https://livestreams.com

pre-transcode) and store several versions of the same video [10]. Given the number and volume of video repositories in current SSPs, pre-transcoding and storing imply maintaining massive storage and processing resources. However, recent studies (*e.g.,* [16]) show that the access pattern to video streams in a repository is not uniform and follows a long-tail pattern. That is, many video streams are rarely accessed by viewers, and only a small portion of them (generally known as *hot video streams*) are accessed frequently by viewers. Therefore, SSPs have become motivated to pre-transcode only hot videos and transcode the rest of them upon users' demand. However, this approach requires a large computational infrastructure (or datacenters) to process video streams in a real-time fashion.

Provisioning and upgrading built-in infrastructures to meet these demands is cost-prohibitive and distracts SSPs from their mainstream business, *i.e.,* producing video content and focusing on viewers' satisfaction. Therefore, SSPs have become extensively reliant on cloud providers to provide their services [9]. Cloud providers offer abundant of reliable computational and storage services to SSPs. Making use of cloud services, however, has introduced new challenges to SSPs. In particular, their challenge is to minimize the incurred cost for renting cloud services, while maintaining QoS for their viewers.

To overcome these challenges, several research works have been undertaken in estimating the cost of using cloud services when they are used in different steps of video streaming. In particular, there are studies on video transcoding time [17, 9], video segmentation models [18, 10], scheduling [19, 17], and resource provisioning methods [10, 20]. However, these studies generally focus on a portion of the video

streaming process and how that portion can be performed efficiently on the cloud.

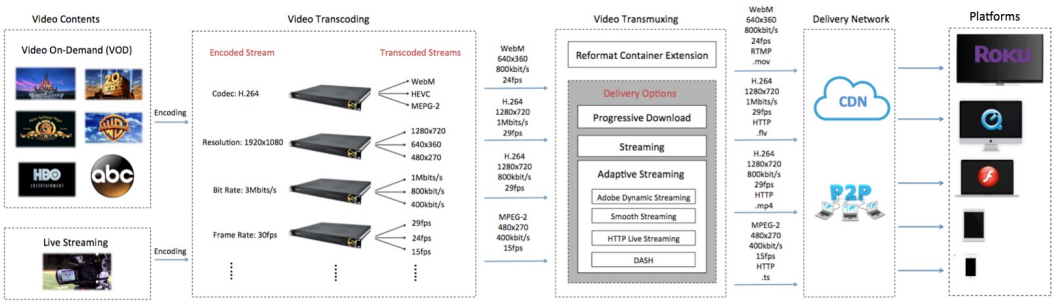
Further studies are required to combine these works in a bigger frame and provide a higher level vision on the efficient use of cloud services for different steps of video streaming. This is also important because currently, cloud providers offer various types of VMs (*i.e.,* heterogeneous VMs). For instance, Amazon EC2 offers General, CPU Optimized, and GPU VM types with different architectural characteristics and diverse costs. In such a Heterogeneous Computing (HC) environment, different transcoding tasks can potentially have various transcoding times (*i.e.,* execution times) on the heterogeneous VMs. That is, different video transcoding operations may have different affinities with different VM types. In this study, we term the affinity between cloud VM types and transcoding tasks as *task-machine affinity*. Currently, there is no study of this kind available.

In this chapter, we provide a detailed survey of the research works undertaken on the use of clouds for video streaming. However, before that, we first explain in detail the structure of video streams and elaborate on steps that need to be done for video streaming. Then, we will explain the computational or networking services demanded at different steps of video streaming. Next, we explain how clouds can fulfill these demands and the research works undertaken for that purpose. In the end, we discuss emerging research areas to enable video streaming in future.

2.2. How Do Video Streaming Services Work

To stream a pre-recorded or live video, it has to go through three main modules: *video encoding/transcoding, video transmuxing/packaging,* and *Delivery Network*, as shown in Figure 2.1 Most devices can take HD or even 4K video contents recently. A raw filmed or

Figure 2.1. A workflow of video streaming.



recorded video content may need GBs or TBs of storage, which makes it difficult to store, not to mention streaming it over the Internet. However, the video just continuously shows a group of pictures at a certain rate. If there are 25 pictures in one second, it must contain a lot of redundancies. To remove these redundancies, that is why we have to encode and compress the video to a much smaller space with one of the available compression standards, *e.g.,* H.274/AVC [21], VP9 [22], and HEVC [23] *etc.* Based on these standards (codec), the original videos are encoded to a specific resolution, bit rate and frame rate that can be played or streamed to devices.

However, considering the large, diverse devices on the market and different Internet access ability of the viewers, a video with only one specific codec, resolution, bit rate and frame rate cannot suit for all. To be able to meet such diverse demands, an encoded video usually has to be transcoded to different versions, to produce multiple codec, resolution, bit rate and frame rate videos. Video transcoding will be discussed in Section 2.3.3. in more detail.

With all different transcoded renditions, the video has no problem being played on viewers' devices. However, in order to stream the video through the Internet, we have to obey different streaming protocols, *e.g.,* HTTP [24], RTMP [25], and RTSP [26] *etc.*.

Therefore, we have to wrap the encoded video content with some container formats, *e.g.,* ISO BSMFF or 3GPP TS. Inside these container format headers, it specifies the supported streaming protocols and the rules to send video chunks. The whole process is called video transmuxing/packaging, and more details will be discussed in Section 2.3.4.

After transmuxing, a video can be streamed and played whenever it is requested by viewers. However, due to the limitation of bandwidth, if there are too many incoming requests for the same videos, it can congest the network and burden to the server. To reduce the request numbers to the server and also provide fast startup time, a content delivery network (CDN) [27] is utilized to cache video contents at edge servers near viewers at different geological locations. Another option is no central servers, but utilizing each receiver (*e.g.,* viewer's computer) as a server as well to send video contents to another receiver, which is called P2P [15]. More details of the delivery network will be discussed in Section 2.3.5.

With the development of video streaming, other topics like video content security, analyzing video views and efficient storage become important as well. They will be discussed in Section 2.3.7., Section 2.3.8. and Section 2.3.6., respectively.

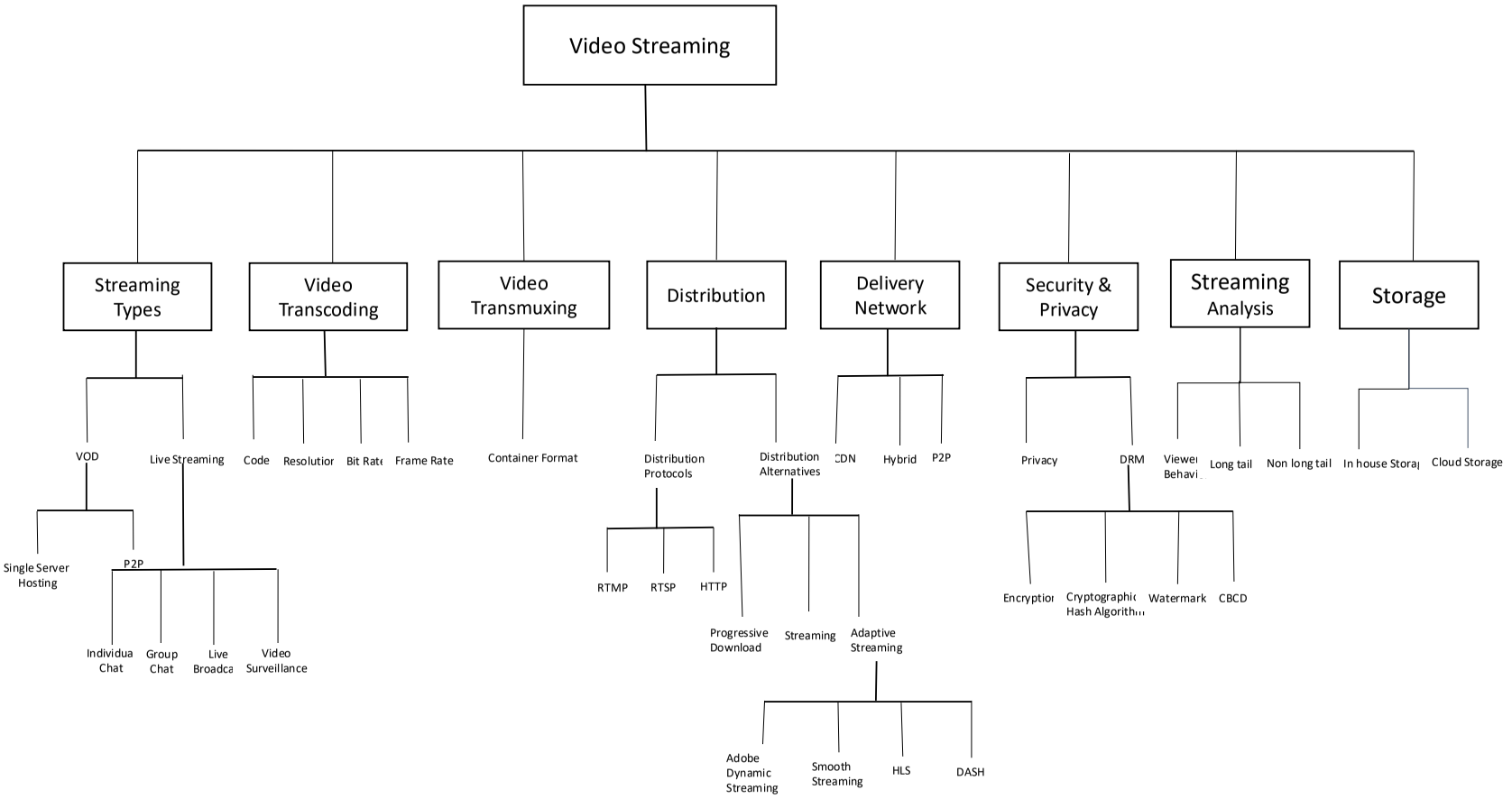
2.3. Challenges and Issues on Video Streaming

2.3.1. Overview

Figure 2.2 presents a taxonomy of the video streaming phases and issues that video streams go through when they are requested by a user. In this chapter, we discuss video streaming types (*i.e.,* VOD and live streaming), then we brief the types of video transcoding and video trunsmuxing. For the distribution layers of video streaming, we summarize the major protocols. We also explain some technologies that are used to

stream videos such as CDN and P2P. It is crucial to address security and privacy issues for video streaming such as encryption, cryptographic, watermark, and CBCD. In addition, we discuss viewer behaviors and access patterns to videos streams. Last but not least, we discuss the storge types where video streams are stored such as in-house storage and cloud storage.

Figure 2.2. A taxonomy of video streaming



2.3.2. Video Streaming Types

We consider two main categories of video streaming based on the video origin and time of recording: video-on-demand (VOD) and live-streaming. Video-on-demand (VOD) describes a scenario in which an archived video that was previously recorded is streamed to another computer, while live streaming involves a video that is not archived and is recorded at the time of streaming. While the end-user experience may be similar for each, here are several important differences in their implementation.

Both categories have similar cardinalities, though what the cardinality means can differ. VOD is primarily done in a one-to-many context, in which the video file is hosted on a single server and is streamed to many viewers (*e.g.,* YouTube). These systems, however, incur large costs to the server. This can be remedied using a many-to-many, peer-to-peer filesharing approach to streaming [28].

Live-streaming is most commonly done through a peer-to-peer framework to cut out the need for an external server. Different types of live-streaming can be further categorized by the quantity of streamers and viewers. The simplest case is one-to-one streaming, which occurs in a two-person video chat (*e.g.,* Skype). Similarly, many-to-many streaming can occur in group chats consisting of more than two people. One-to-many streaming occurs when a single camera or source streams to many viewers, (*e.g.,* live television broadcasts). Finally, many-to-one streaming describes many cameras streaming to a single device (*e.g.,* a video surveillance system).

The experience of each streaming category is heavily affected by buffer sizes [29]. Because videos-on-demand are archived, and there is typically no desire to keep the time in sync with another participant, the buffer size can be as large as the viewing hardware

will allow. This allows high quality frames to be built up and leaves remaining frames if there is a drop in service on either end. Live-streamed video typically demands the viewer to keep within a small number of seconds of the streamer, thus demanding a small buffer size. For one-to-many communication, such as live event broadcasting, several seconds of delay are often tolerated, as the communication is only going one-way. However, for streaming that involves two-way communication (individual or group chat), latency must be reduced to the smallest amount possible to maintain natural human communication with minimal interruptions. This requires a very small buffer, so that frames are displayed more quickly, and often necessitates lower quality video.

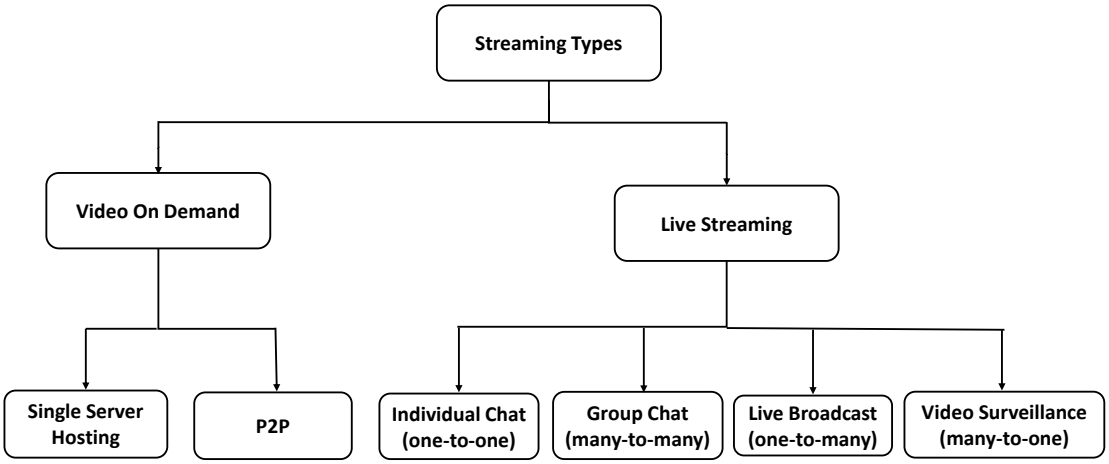
Another unusual type of video streaming that has recently increased in popularity is a method we will term *asynchronous interactive video streaming*. This method is used for applications such as online games or video processing [30], which may be performed on machines that are ill-equipped to handle the processing load required by the application but can receive a video stream. Using this architecture, a client will send their input data to a cloud server supplied by the service provider, which will perform any necessary input handling and parsing, update the interactive graphics, and send the video to the client as a stream. The result, to the client, is similar to an interactive live stream.

In summary, Fig. 2.3 shows a taxonomy of the video streaming types. Each type of streaming faces its own challenges which are described in the coming sections.

2.3.3. Video Transcoding

A video stream, as shown in Figure 2.4, consists of several sequences. Each sequence is divided into multiple *Group Of Pictures* (GOP) with sequence header information in the beginning of each GOP. A GOP is essentially a sequence of frames beginning with an `I`

Figure 2.3. A taxonomy of streaming types considered by this work

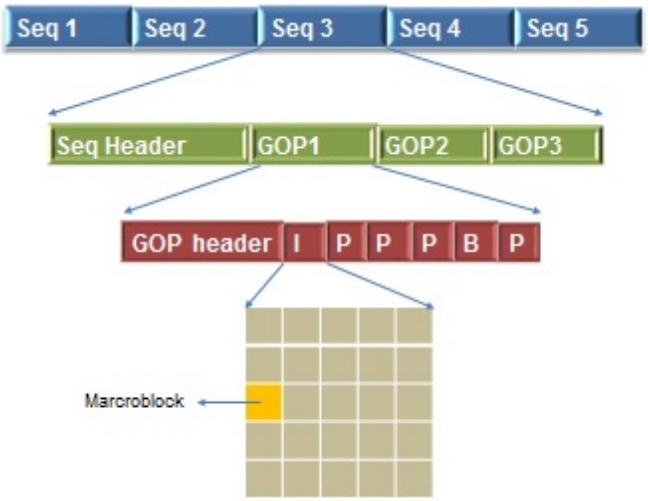


(intra) frame, followed by a number of `P` (predicted) frames or `B` (be-directional predicted) frames. Each frame of the GOP contains several *slices* that consist of a number of *macroblocks* (MB) which is the unit for video encoding and decoding operations.

The video transcoding process can operate at different levels, namely sequence level, GOP level, frame level, slice level, and macroblock level. However, transcoding is commonly carried out at the GOP level [18, 31] because each GOP can be processed independently. Similarly, in this work, we assume that all the transcoding processes operate at the GOP level.

Video contents are initially captured with a particular format, spatial resolution, frame rate, and bit rate. Streaming service providers usually have to adjust the original video based on the client's network bandwidth, device resolution, frame rate, and video compression standard (*i.e.,* codec). These conversions are carried out on all GOPs of a video and are termed *video transcoding* [6, 32]. Below, we provide more details on the

Figure 2.4. The structure of a video stream that consists of several sequences. Each sequence includes several GOPs. Each frame of GOP contains several macroblocks.



nature of processing in different transcoding operations:

Bit Rate Adjustment

To stream high quality video contents, the videos are encoded with a high bit rate. However, high bit rate also means the video content needs a larger network bandwidth for transmission. Considering the diversity and fluctuations of network bandwidth on the viewer's side, streaming service providers usually need to change the bit rate of video streams to ensure smooth streaming [33].

Spatial Resolution Reduction

Spatial resolution indicates the dimensional size of a video. The dimensional size of an original video stream does not necessarily match the screen size of viewers' devices. Thus, to avoid losing content, macroblocks of an original video have to be removed or combined (*i.e.,* downscaled) to produce lower spatial resolution video. There are also circumstances where the spatial resolution algorithms can be applied to reduce the spatial resolution

without sacrificing quality [34].

Temporal Resolution Reduction

Temporal resolution reduction happens when the viewer's device only supports lower frame rate. In this situation, the streaming service provider has to drop some frames. Due to the dependency between frames, dropping frames may cause motion vectors (MV) to become invalid for the incoming frames. Temporal resolution reduction can be achieved using methods explained in [35].

Video Compression Standard Conversion

There is a wide variety of video compression standards (codec) for video files —from MPEG2 [36], to H.264 [37], and to the most recent one, HEVC [23]. Without these compression standards in place, the video size would be too large and cannot be streamed or even stored using the current network and storage capacities.

Viewers' devices usually support only one or few compression standards. Hence, if the video codec is not supported on the viewer's device, then the video needs to be transcoded based on the supported codec on the viewer's device [38].

2.3.4. Video Transmuxing/Packaging

To transmit a pre-encoded video file from server to viewer involves multiple layers of network protocols [39], namely physical layer, data link layer, network layer, transport layer, session layer, presentation layer, and application layer.Those protocols define the syntax of stream file headers, data, authentication and error handling. All streaming protocols are in the application layer, which means they can use any layer beneath it to transmit data packets.

Choosing the right streaming technology involves understanding the benefits and drawback of the accompanied streaming protocols and file formats. In this chapter, we will mainly discuss three streaming technologies in the market, namely *Progressive Download, Streaming,* and *Adaptive Streaming.*

Progressive Download

In the past, a video could not be viewed until it was completely downloaded, which means users usually have to wait minutes or even hours to start watching the video. Progressive downloading resolves this issue by allowing video to be played as long as the player's initial buffer is loaded instead of waiting for the whole video to be downloaded. This reduces the waiting time down to 3-10 seconds to begin to watch a video [40].

However, due to the downloading feature, progressive downloading may face two problems. Since the video is downloaded linearly, if the bandwidth is too low, the user cannot move forward a video too much until that part has been fully downloaded. On the other hand, if a video file is fully downloaded, but the user stops watching in the middle, the rest of the video bandwidth is wasted, not to mention the copyright issue since the whole video is downloaded on the user's hard drive. Also, progressive downloading is utilizing HTTP protocol [24] which is commonly used in a web application, sending HTML text, GIF and JPG graphics, PDF files *etc.*to remote viewers. HTTP protocol is based on TCP transport protocol [41], which provides better reliability and error-resilience than UDP [42], while it also increases the network latency [43] which is essential for live streaming. These drawbacks of HTTP-based progressive downloading pushed the development of streaming technology.

Streaming

Unlike progressive download which delivers video contents through HTTP servers, streaming technology uses real-time protocol (RTP) [44] and delivers video contents from a separated streaming server. These streaming servers work with traditional HTTP servers. Whenever a viewer clicks a link on the HTTP server, it initiates a connection between the streaming server and the player that persists until the viewer stops watching. Because of this connection, these protocols are considered stateful, as compared to HTTP, which is stateless and has no connection between server and player.

Thanks to this connection, streaming protocols allow random access in the video file, as well as adaptive streaming, where multiple encoded video streams could be delivered to different players based upon available bandwidth and CPU power. The streaming server can also monitor outbound flow, so if the viewer stops watching, it stops sending video packet to the viewer, and little extra bandwidth will be wasted. Video contents in the streaming server are split into small chunks. Whenever these chunks are sent to the viewers, they are cached at the local and will be removed after they are played. This feature enables quickly changing play locations but also protects the copyright of the video content.

The shortcomings of streaming technology, however, are noteworthy too. A streaming protocol like RTMP [25], which is used by Adobe flash, utilizes different ports from HTTP. RTMP packets may be blocked by certain firewalls, though the Adobe Media Server has workarounds if these problems are experienced. The consistent connection between the streaming server and viewer players increases more bandwidth cost. The scalability of the streaming server is limited while there are much more HTTP servers

available out there.

These doubts aside, there is a general perception that HTTP-based technologies are more effective at delivering high-quality streams.

Adaptive Streaming

As mentioned, several innovations in HTTP streaming also addressed previous limits of the technology. As before, there is no persistent connection between the server and the player; the video resides on any HTTP server, and the technology remains stateless. However, now all HTTP-based streams are broken into chunks, either separate files or segments within a larger file. Rather than retrieving a single large file with a single request, HTTP-based technologies retrieve consecutive short chunks on as needed basis.

This has multiple benefits. First, there is little waste because the video is delivered as it is watched. This effectively meters out the video, enabling a single HTTP server to efficiently serve more streams. Since these technologies are delivered via HTTP, they sidestep the issues faced by RTMP. HTTP-based technologies are firewall friendly and can leverage HTTP caching mechanisms. Because no streaming server is required, they are less expensive to implement and can scale more cheaply and effectively to serve available users. Seeking is no problem; if the viewer drags the player head forward, the player can just retrieve the appropriate chunks. These technologies also enable the efficient switching between streams, so called adaptive streaming.

There are four main adaptive streaming protocols, namely MPEG's Dynamic Adaptive Streaming over HTTP (DASH) [45], Adobe's HTTP Dynamic Streaming (ADS) [46], Apple's HTTP Live Streaming (HLS) [47], and Microsofts' Smooth Streaming [46].

DASH defines a Media Presentation Description (MDP) and extensions to the well-known ISO Base Media File Format (ISOBMFF) [48]. The former is an XML document providing a manifest/session description which enables the client to request individual media segments via HTTP. The media segments are compliant with a delivery format that has been derived from the ISOBMFF [49].

HDS is based on Flash media manifest and F4F file format. The former is an XML document similar to 3GPPs MPD and the latter are MP4 fragment files, *i.e.,* also based on ISOBMFF. However, the solution is proprietary and not compliant with 3GPP AHS.

HLS is well known for quite some time and implemented on the apple devices and nowadays android, too. It utilizes an M3U playlist file which serves as the manifest, and each media file must be formatted as an MPEG-2 Transport Stream (M2TS) [50].

Smooth Streaming has also existed for a while which utilizes a server manifest file (*i.e.,* SMIL document) and a client manifest file (*i.e.,* proprietary XML document). Furthermore, this approach defines a smooth streaming format (ISMV) as an extension of the ISOBMFF.

Table 2.1. Adaptive Streaming Supported Platforms

	Desktop Player	Mobile Device Support	OTT Support
MPEG-DASH	dash.js, dash.as, GPAC	Windows, Android Phone	Google TV, Roku, Xbox 360
HTTP Dynamic Streaming (HDS)	Flash, AIR	Android/iOS(via AIR app)	None
Apple HTTP Live Streaming (HLS)	iOS, Mac OSX, Flash	iOS/Android3.0+	Apple TV, Boxee, Google TV, Roku
Smooth Streaming	Silverlight	Windows Phone	Google TV, Roku, Xbox 360

2.3.5. Delivery Network

Delivery Network

Content Delivery Network (CDN) services: CDN is a technology that reduces the delay to access different static content types, including video streams, through the Internet. CDN technology replicates the content (*e.g.,* video content) in different geographical areas to minimize the network travel time of content to users [51, 52]. Some VSPs turn to CDN to deliver their videos streams to users with better quality and minimum delay [53, 54]. Netflix uses CDN to stream their movies to end-users with low latency and high quality. The platform of Netflix employs three CDN providers (Akamai, LimeLight, and Level-3) to cover all regions of users. The same video and same quality are delivered from the three CDNs [14].

Moreover, streaming videos through CDNs has been addressed in earlier works [55, 56, 57], Cranor *et al.* [56] proposed an architecture (PRISM) for distributing, storing, and delivering high quality streaming content over IP networks. Their design offers a framework and architectural components necessary to support video-on-demand streaming content distribution networks (CDN) services. Wee *et al.* [57] presented an architecture for mobile streaming CDN which was designed to fit the requirements of mobility, wireless and scaling issues.Moreover, CDN can be enhanced by merging with systems to improve its performance. Apostolopoulos *et al.* [55] proposed Multiple Description Streaming system to provide diversity paths between nearby edge servers and clients in order to reduce latency and deliver high quality streaming.

Live streaming CDN is discussed in [58] in terms of scalability, quality, and reliability. In addition, several previous works are based on live streaming and P2P

CDN [59] to present better quality, more scalability, reliability, and low latency.

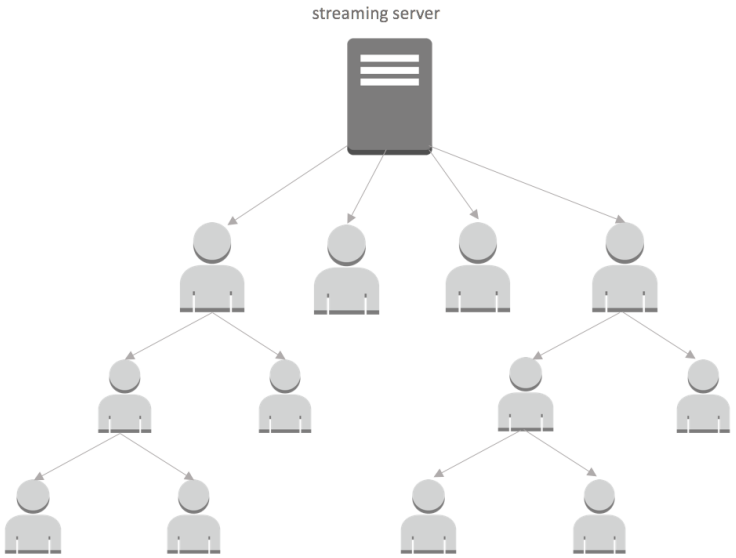
On the other hand, CDN involving video transcoding consists of a storage cloud which aims for storing transcoded video streams, a transcoding cloud which includes of a set of servers to run transcoding tasks, and edge servers to deliver transcoded video streams to users. Edge servers are deployed in numerous data centers to cover all geographical regions [60, 61].

Although there are many advantages of using CDN to stream videos, there are still some disadvantages of using it, for instance, the high cost and less scalability with maintaining the Quality of service QoS.

Peer to Peer (P2P)

Peer-to-peer (P2P) is a network architecture based on the operations of distributed computing systems. They enable the direct sharing of computing resources (*e.g.,* CPU cycles, storage, content, etc.) among peers (nodes) in the network [15]. Peer-to-Peer (P2P) networking is designed for both clients and servers to act as peers to download the data from the network and upload the data to other users in the network. P2P aims to quickly disseminate the data files among users in less time. P2P technology has been utilized in video streaming services. Several P2P streaming systems have been designed to support on-demand and live video streaming [62]. P2P streaming is categorized into two types tree-based and mesh-based. P2P tree-based structure distributes video streams by sending data from a peer to its children peers. In P2P mesh-based structure, the peers are not related to a specific topology; this peering structure instead functions based on the content and bandwidth availability of peers [28]. One drawback of using tree-based streaming is the vulnerability to peer churn. The disadvantage of deploying mesh-based

Figure 2.5. Single-tree based streaming

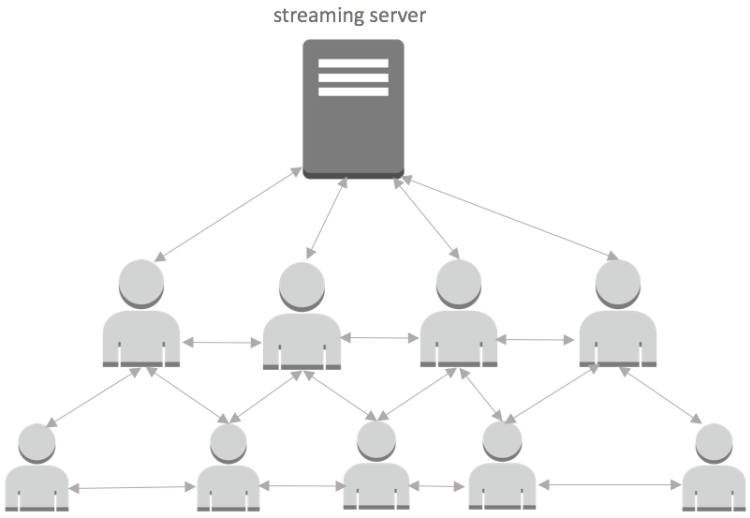


streaming structure are video playback quality degradation ranging from low video bit rates, long startup delays, to frequent playback freezes. P2P streaming systems attain immense triumph due to high scalability, low cost and robustness to churn.

In *Single Tree-based streaming* Figure 2.5, the carrying out of video streaming is done at the application layer, and the streaming tree is formed by the participating users. Each user joins the tree at a certain level. It receives the video from its parent peer at the level above, and forwards the received video to its children peers at the level below [63, 64].

In *multi-tree (mesh-tree) streaming* Figure 2.6, the user (peer) establishes peering relationships with multiple neighboring peers. A peer may download/upload video from/to multiple neighbors simultaneously. If a peer's neighbor leaves, the peer can still download video content from remaining neighbors. At the same time, the peer will find new neighbors to keep a desired level of connectivity. The high peering degree in Mesh-based streaming systems makes them extremely robust against peer churn [65, 66].

Figure 2.6. Mutli-tree based streaming



Tree-based P2P video-on-demand streaming shares the video stream on the network and achieves fast downloading by swarming. The video stream is divided into a small data block. The server disperses the data blocks to different users. The users download from its neighboring peers the blocks that they currently do not have [67]

Guo *et al.* [68] proposed an architecture that uses a peer-to-peer approach to cooperatively stream video using patching techniques, while only relying on unicast connections among peers. Xu *et al.* [69] proposed a Balanced Binary Tree-based strategy for Unstructured video-on-demand distribution in P2P networks (BBTU). BBTU assumes videos can be divided into several segments which can be fetched from different peers. Yiu*et al.* [70] propose VMesh, a distributed peer-to-peer video-on-demand (VoD) streaming scheme which efficiently supports random seeking functionality. In VMesh, videos are divided into segments and stored at peers' local storage in a distributed manner. An overlay mesh is built upon peers to support random forward/backward seek, pause and restart during playback. Cheng *et al.* [71] proposed a novel ring-assisted overlay

topology, called RINDY, to efficiently support VCR functions for VoD services in such networks. In RINDY, a peer can implement fast relocation of random seeks by maintaining some near neighbors and remote neighbors in a set of concentric rings with power-law radii. Xu *et al.* [72] propose a derivative tree-based overlay management scheme to support user interactivity in the P2P streaming system. The derivative tree takes advantage of well-organized buffer overlapping to support asynchronous user requests while brings high resilience to the impact of VCR-like operations.

The main advantages of P2P are low cost and less flexibility for scalability with the instability of QoS. Therefore, researchers thought to combine the benefits of both CDN and P2P to be in one system. Thus, this leads to a hybrid system that combines P2P and CDN for content streaming. Several works have addressed hybrid P2P and CDN networks, Gamerhi *et al.* [73] proposed a novel replication strategy which enables traditional CDNs to offer Hybrid CDN-P2P streaming content delivery services. The proposed solution relies on the economic model of the Hybrid CDN and employs a dynamic mechanism to optimize the number and places of replicas for P2P service. Xu *et al.* [74] proposed a novel hybrid architecture for that combines both CDN and P2P based streaming media distribution. The design is highly cost-effective: it significantly lowers the cost of CDN capacity reservation, without compromising the media quality delivered.

2.3.6. Storage

Storage challenges

Many challenging issues face storage system design for multimedia, high throughput, large capacity, and fault tolerance. When the video is stored on one disk, the concurrent accesses to that video are limited by the throughput of the disk; this limits the number of

viewers to the video at the same time. Shenoy *et al.* [75] proposed data stripping where a video is segmented and saved across multiple storage disks. This extends the throughput of storage disk and overcomes the limitations of viewers at the same time.

Videos streams are segmented into blocks when storing on disk. The blocks can be stored one after another (contiguously) or scattered over the storage disks. Contiguous storage method is simple, but it comes with problems of fragmentation and imposes copying overhead during insertions and deletions. The scattered method eliminates fragmentations problems and copying overheads. Considering video streams storage on one single disk has throughput limitation. However, multiple disks storage are used to increase the throughput at the cost of additional storage spaces. Video streams are scattered across the various disks and can be implemented by data striping and data interleaving[76]

Video streaming storage becomes also challenging with the diversity of users devices. Many versions of a single video should be stored to meet the user's specifications. Therefore, storing all those video versions requires large-scale storage space and imposes constraints to store all video versions. Previous studies addressed techniques to overcome storage issues of video streaming. Miao *et al.* [77] proposed algorithms to store some frames of video on proxy cache because the proxy cache is space limited. Their proposed algorithms on proxy cache contribute to reduce the network bandwidth costs and to increase the robustness of streaming video poor network conditions happens such as delay, congestion, and loss.

2.3.7. Security

Privacy

Privacy, in the context of video streaming, is primarily a concern when considering video surveillance. The widespread increase in the use of video surveillance has led to a rise in concerns about the privacy of those being watched, on both personal and legislative levels. Because video surveillance is necessarily invasive, the challenge becomes how to balance the public privacy of individuals with the needs of the surveillance system. An ideal solution would be let protected data (*e.g.,* faces) remain in the video feed while obscuring personally identifying features.

The majority of solutions to the privacy problems utilize computer vision to eliminate privacy revealing information. For example, a Sony patent [78] provides a solution that detects a person in frame's skin tone and replaces it to obscure racially identifying information. Another solution by Senior *et al.* [79] presented an architecture for a system that would encode and encrypt incoming video streams at different levels of privacy protection for different levels of user clearance. For example, a low-clearance-level user may only see a person in the stream as a rough outline. Techniques using a layered access approach are conventional, as high-level users will still need to see identifying information if, *e.g.,* crime is committed. Other solutions still use warping techniques to eliminate privacy-revealing information on watched persons' faces [80].

2.3.8. Analysis

Impact of quality on viewer behavior

Previous studies in [81] showed that the quality of video stream has an impact on viewers behavior regarding abandonment the video. When a video stream starts up with delay,

the viewers abandon watching the videos. With incrementing the startup delay, the percentage of views who abandon video watching increases. Interruptions during video playing reduces the number of viewers who access the site of video streaming providers. Low-quality videos have fewer revenues for video streams providers.

Florin *et al.* [82] addressed the impact of video quality on user engagement. They claimed that The percentage of time spent in buffering (buffering ratio) has the most significant impact on the user engagement across all types of content and the live stream is most impacted when the number of buffering in a video increase. The average bitrate of live streaming has a significant impact on user abandonment of watching the video. Startup latency and the number of buffering during video playing reduce the engagement of users. These quality issues reduce the revenues to video streams providers. Therefore, video streaming providers attempt to maximize user engagement by minimizing the buffering time and rate and increase the average bitrate.

Access Pattern to Repository of Video Streams

In a repository of video streams provider, the distribution of videos follows the long tail distribution. The popular videos are accumulated at the beginning of the curve; those videos are very accessed by viewers while the nonpopular videos are located in the long tail of the curve, and they are rarely accessed.

Viewers are more interested in recently posted videos. Thus, the recent videos posted in a period are requested more than old videos. Moreover, the probability to predict the popularity of new videos in the near future are greater than the probability to predict the popularity of old videos. The significance of popularity distribution shift depends on the age of videos. For new videos, the popularity distribution is significant

while the popularity of old videos does not shift too much [83]

Access Pattern to Video Streams

Video access rate indicates the number of times a video is requested, however, it does not determine if the requested video stream is watched to end of it or not. In fact, recent studies reveal that, in a video stream, the beginning GOPs are watched more frequently than the rest of the video stream. Miranda et al. [13] show that the distribution of the views within most of the video streams in a video stream provider follows a long tail distribution. More specifically, they show that the distribution of accesses to GOPs of a video stream can be expressed by the Power-law [84] model.

Although GOP access rate for most of the video streams in a repository follow a long tail pattern, there are video streams whose GOPs' access rate do not support this pattern. In these video streams, some scenes that can be in the middle or end of the video stream is accessed more frequently than other scenes. An example of scenes with tremendously higher access rate can be found in a soccer match where a player scores a goal. We define this type of videos as those with called non-long-tail access pattern [85].

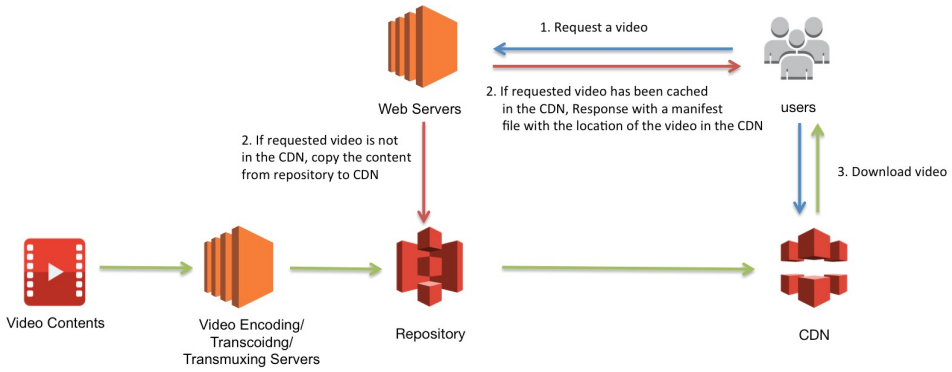
2.4. Cloud-based Video Streaming

2.4.1. Overview

Video streaming has widely utilized cloud services for a different purpose. Due to the scalability of cloud cluster, raw video contents are encoded, transcoded and then transmuxed (packaged) on cloud virtual machines (VMs) (*e.g.,* AWS EC2). Packaged video files are stored in cloud storage (*e.g.,* AWS S3) with very low cost. To reduce the network latency and reduce the burden of a request for the server, video contents are also cached in the CDN (*e.g.,* CloudFront) where geographically close to viewers. When a

viewer requests a video, a request is sent to a web server which also in the cloud cluster. If the requested video has been cached in the CDN, the web server will send back a manifest file with the location of the video files or segments in CDN, so that viewer can send another request to CDN and download videos from there. However, if the requested video is not in the CDN, web server has to copy the content from cloud storage to CDN, and then send back the manifest with the new video file location in CDN. A workflow of the whole request until download video process is shown in Figure 2.7. The rest of this section, we will discuss how does cloud services impact on different layers video streaming.

Figure 2.7. The workflow of video streaming



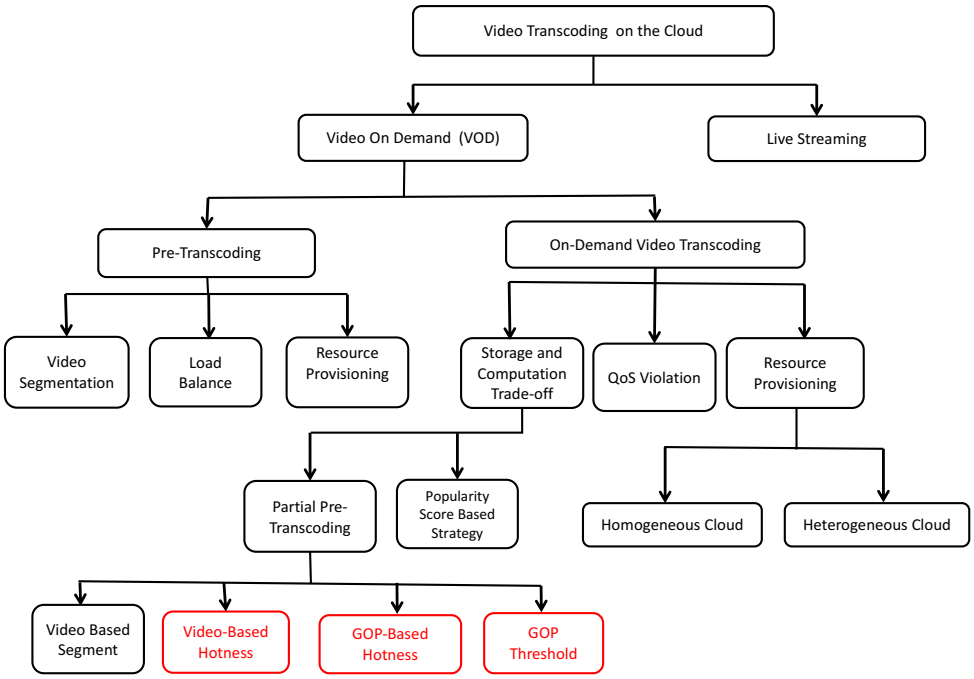
2.4.2. Cloud-based Video Transcoding

Video transcoding is an expensive and time-consuming operation. Techniques, architectures, and the challenges of video transcoding were investigated by Ahmad *et al.* [6] and Vetro *et al.* [32]. With the rise of cloud computing, SSPs realize a more cost-efficient way to transcode videos by utilizing cloud services.

A taxonomy of the studies undertaken on cloud-based video transcoding is illustrated in Figure 2.8. Challenges of cloud-based transcoding for VOD was studied in [20, 19]. Studies have been concentrated on both pre-transcoding [86, 18, 87, 20, 19],

on-demand transcoding [10, 88, 89] and live streaming [9, 90, 91].

Figure 2.8. A taxonomy of researches undertaken on video transcoding using cloud services. Red blocks position the contributions of this dissertation



For pre-transcoding, the research focus are mainly on video segmentation [86, 18], load balance [87, 19], and resource provisioning [87, 20] *etc.*, while quality of service (QoS) is not a concern because different versions of the same video will be ready before releasing to viewers. However, transcoding the whole repository videos into multiple versions and storing all the versions causes massive storage cost for SSPs.

To reduce the storage cost while remaining QoS, on-demand video transcoding has been proposed in [10, 88, 89]. Li *et al.* [10] propose a CVSS architecture with a startup queue which prioritizes the beginning part of each video to proceed to the transcoding servers. With proper scheduling and resource provisioning policy, it provides low startup delay and playback jitter. Li *et al.* [89] present a Cloud Transcoder which utilizes an

intermediate cloud platform to bridge the format/resolution gap for mobile devices in real-time. It only requires the user to upload a video request with specified transcoding parameters rather than the video content. Cloud Transcoder transcodes downloads and transcodes the original video on the user's demand and deliver the transcoded version the user.

Transcoding video in an on-demand way does reduce the storage cost, whereas it incurs computing costing, how to balance these two costs becomes another challenge. Jokhio *et al.* [92] presents a computation and storage trade-off strategy for cost-efficient video transcoding in the cloud. The trade-off is based on the computation cost versus the storage cost of the video streams. They determine how long a video should be stored or how frequently it should be re-transcoded from a given source video. Zhao *et al.* [93] take the popularity, computation cost, and storage cost of each version of a video stream into account to determine versions of a video stream that should be stored or transcoded. Also, Zhao *et al.* [94] extended his earlier work and implemented his model on segments within a video. Compared with other methods, his extended work reduces additionally the total incurred cost. Darwich *et al.* [95] proposed a method to partially pre-transcode a video stream with long access. Our method finding the first GOP (*e.g.,* GOP threshold) that has its cost ratio greater than 1, GOPs which are located before GOP threshold are pre-transcoded, and GOPs which are located after GOP threshold are re-transcoded. Thus, GOP threshold method reduces the cost of video transcoding using clouds services. Our method is efficient for a repository that contains video streams with long tail access. Krishnappa *et al.* [61] proposed transcoding policies that transcode video segments that are requested by users. To maintain the quality of videos regarding startup delay (or

playtime delay) when applying online policies on video, they proposed a method to predict the next video segment that is requested by a user. They implemented their prediction model by using Markov theory. Their proposed method shows a high reduction in the cost of using cloud storage with high accuracy.

The idea of cloud-based video transcoding has also has been applied to live video streaming [9, 90]. Timmerer*et al.* [96] present a live transcoding and streaming-as-a-service architecture utilizing cloud infrastructure taking the live video streams as input and output multiple stream versions according to the MPEG-DASH [91] standard. Lai *et al.* [90] design a cloud-assisted real-time transcoding mechanism based on HLS protocol, implement the bandwidth recorder, segment transrater, and segment redirector on the server, instantly analyzes the online quality between client and server without changing the HLS server architecture and provides the optimum media quality.

With the trend of video transcoding using cloud service, better understanding the performance of different video transcoding operation on different VM type will be helpful. Transcoding time estimation plays an important role in both efficient scheduling and resource provisioning. Deneke *et al.* [17] utilize machine learning based on the video characteristics (*e.g.,* resolution, frame rate, bit rate *etc.*) to predict the transcoding time which results in better load balance for transcoding jobs. Seo *et al.* [97] focus more on the transcoding process details to estimate transcoding time, such as discrete cosine transform (DCT), inverse DCT (iDCT), quantization (Q), inverse Q (iQ), motion estimation/motion compensation (ME/MC), variable length coding (VLC), variable length decoding (VLD). While both [17, 97] do not consider the diversity of heterogeneous environment of cloud services. Our work provides a deeper performance analysis and transcoding time

estimation for different transcoding operations on different VM types, which will be beneficial for later video transcoding scheduling and resource provisioning by utilizing heterogeneous cloud services.

Several studies explored the performance analysis of heterogeneous cloud services [98, 99, 100, 101, 102]. Iosup *et al.* [98] and Jackson *et al.* [100] studied application oriented performance analysis using heterogeneous cloud services. The results show that although cloud services have their drawbacks in terms of communication and processing, utilizing cloud services is a viable solution for processing big scientific data workloads, especially those need resources instantly and temporarily. Also, they show that the overall performance drops for applications with long communication requirements. Lee *et al.* [99] further investigated the task-machine affinity in heterogeneous clusters. They propose a shared metric in the heterogeneous cluster to provide a scheduling method that considers fairness. However, there is no study in the literature that focuses on executing video transcoding tasks on heterogeneous VM types in clouds.

2.4.3. Cloud-based Video Transmuxing/Packaging

Video Transmuxing (Packaging) is much less complex than video transcoding, therefore it does not take too much to process on the servers. For VOD service, video content is usually transcoded different renditions, and then each rendition will be packaged to different versions to meet different streaming protocol requirements (*e.g.,* DASH, HLS, Smooth). This results in too many copies of the original video have to be stored in the repository which raises the cost.

With the flexibility to choose a different type of VMs, as well as the scalability to scale up and scale down VM numbers, cloud service has made it possible for dynamic video

transmuxing [103]. Instead of statically package a transcoded video into different protocol renditions and storing in the repository, dynamic video transmuxing detects viewer's player and its supported protocol and generate a transcoded video with this protocol and directly deliver to that client. The whole process only takes milliseconds [104], which viewers will barely notice, but saves a lot store cost for video streaming providers.

2.4.4. Cloud-based Delivery Network

CDN-Based Video Streaming

Content delivery networks (CDNs) using storage clouds have recently emerged to further reduce the delivering time of video streams to users, and it is known as cloud CDN. Compared to traditional CDNs, storage cloud-based CDNs have the advantage of cost-effectively offering hosting services to Web content providers without owning infrastructure. However, as the only cost for cloud CDNs is the bandwidth and storage cost [105]. Hu *et al.* [106] present a social video replication and user request dispatching mechanism in the cloud content delivery network architecture to reduce the system operational cost while guaranteeing the averaged service latency. Based on viewing behavior measurement, Hu *et al.* [107] investigated the community-driven sharing video distribution problem under cloud CDN, and they proposed a dynamic algorithm to make a trade-off between monetary cost and QoS, their results came with less operational cost while satisfying the QoS requirement.

Jin *et al.* [108] proposed content delivery-as-a-service (CoDaaS), an innovative idea to enable on-demand virtual content delivery service (vCDS) overlays for user-generated content (UGC) providers to deliver their contents to a group of designated consumers. The proposed CoDaaS solution is built on a hybrid media cloud and offers elastic private

virtual content delivery service with an agreed Quality of Service (QoS) to UGC providers.

Li *et al.* [109] proposed partial migration of VoD services into the hybrid cloud-assisted deployment, where the user requests are partly served by the self-owned servers and partly served by the cloud. Proposed migration strategies (active, reactive and smart strategies) can make the hybrid cloud-assisted VoD deployment save up to 30% bandwidth expense compared with the Clients/Server mode. Some researchers from Microsoft conducted a CDN experiment in Microsoft public cloud, Windows Azure, to demonstrate the benefits of CDN integration with the cloud. The results show a significant gain in large data download by utilizing CDN in Cloud Computing [110].

2.4.5. Cloud Storage

With rapid spreading out videos on mobiles devices, this requires large-scale systems for storage. For this reason, the current storage servers face scalability and reliability problems in addition to the high cost of storage hardware. This also imposes maintenance needs and requires expertise people to deploy and configure the storage servers. The cloud storage comes as solution for scalability, reliability, and fault tolerance [111] Gao *et al.* [112] proposed a scheme that partially transcodes video contents in the Cloud. Their approach aims to store the first segments of video contents which are more frequently viewed, while transcode the remaining video contents online when requested, resulting in storage and transcoding computation cost efficiency. Their system model is based on a partial transcoding of video segments and a dynamic storage system that is adapted to the video access rate change. They designed online algorithm to optimize the performance of their scheme. They stated that the viewers play 20% of the video duration. They adopted the video access pattern by implementing the truncated exponential distribution

to represent the video view rate. They demonstrated that their method reduces 30% of operational cost by compared to storing the whole video.

Zhao *et al.* [113] proposed an approach to do a trade-off between computation and storage costs and to minimize the cost of multi-version videos. They utilized the transcoding weight graph which is the transcoding relationships among versions of a video, along with the popularity of those different versions of the video. Based on the popularity and transcoding relationships among different video versions, their method decides which versions of a video should be stored in the repository or re-transcoded on demand. Their results show reduction in the cost when compared to storing all versions. Also, Zhao *et al.* [94] extended his earlier work and implemented his model on segments within a video. Compared with other methods, his extended work reduces additionally the total incurred cost.

Jokhio *et al.* [92] developed a strategy to strike a trade-off between the computation and storage costs of a video. They estimated the computation cost, the storage cost, and the video popularity information of individual transcoded videos and then utilized this information to make decisions on how long a video should be stored or how frequently it should be re-transcoded from a given source video. They compared their proposal to semisynthetic and realistic load patterns. Their results indicated that their strategy is more cost-efficient than the two intuitive strategies.

In our earlier work [95], we proposed a method to partially pre-transcode a video stream with long access. Our method finding the first GOP (*e.g.,* GOP thresh-old) that has its cost ratio greater than 1, GOPs which are located before GOP threshold are pre-transcoded, and GOPs which are located after GOP threshold are re-transcoded.

Table 2.2. Comparison of different technologies for storing video stream

	Accessibility	Capacity	Scalability	Reliability	Cost	QoS
In House Storage	limited	large	no	no	high	high
Cloud Storage (Outsourcing	high	large	yes	yes	low	low
Content Delivery Network	high	small	no	yes	high	high
Peer 2 Peer	high	large	yes	yes	low	low
Hybrid CDN-P2P	high	small	yes	yes	low	high

Thus, GOP threshold method reduces the cost of video transcoding using clouds services. Our method is efficient for the repository that contains video streams with long tail access.

Krishnappa *et al.* [61] proposed transcoding policies that transcode video segments that are requested by users. To maintain the quality of videos in terms of startup delay (or playtime delay) when applying online policies on video, the propose a method to predict the next video segment that is requested by a user. They implemented their prediction model by using Markov theory. Their proposed method shows a high reduction in the cost of using cloud storage with high accuracy.

In summary, the following table 2.2 briefs video streams storage technologies in terms of accessibility of viewers to the same video stream, capacity, scalability, reliability, cost, and quality of service QoS.

2.4.6. Cloud-Supplemented Video Streaming

The recent vast increase in computational power required to perform tasks on videos (*e.g.,* decoding incoming video, compressing video, editing video, etc...) has inspired researchers to move some of the work to the cloud. Additionally, certain industries are reluctant to give up local legacy servers in favor of a more scalable, but less cost-effective, full-cloud

solution [114]. As a result, several types of research have focused on supplementing existing solutions with the cloud.

One standard manner of using the cloud as a supplement is to use traditional local cluster architecture to deliver content and offload any additional work incurred by high traffic demand to an on-demand cloud system. Zhang *et al.* proposed solutions that would intercept user demands and use statistical analysis to direct the workload to a local or public cloud machine, thus reducing the cloud cost for the service provider [115][114].

If a provider prefers to use a P2P architecture to deliver their content (for reasons of, *e.g.,* lowering infrastructure costs), research has also devised methods for combining cloud with that architecture. For example, Trajkovska *et al.* [116] proposed an architecture with a hierarchy of users, in which primary users would pay more to receive a higher quality stream directly from the cloud. Secondary users would then pay less to receive the stream in a P2P manner from those primary users. Zhao *et al.* [117] further used the cloud to enhance a similar type of P2P network by using the cloud to determine the locality of peers, reducing the incurred inter-ISP traffic.

2.4.7. Container

Scalable cloud servers (VMs) provide the flexibility of video streaming demands while starting a new VM usually takes a few minutes which potentially can impact on the overall quality of service. Researchers and industries usually take a step ahead to predict the demands and begin to launch the VMs earlier.

With the growing of microservices and container technology. Docker container has been widely used in the video streaming workflow. Barais *et al.* [118] propose a microservice architecture to transcode video at lower cost, they treat each module (*e.g.,*

splitting, scheduling, transcoding, merging *etc.*) of transcoding as a separated service and runs them on different docker. Dockers scale up and scale down much faster than VMs, which gives it a big advantage on handling variable streaming demands. Also, docker has been used in video transmuxing/packaging, web service to handle video requests, the advertisement inserting between streams *etc.*

2.5. Summary

In this chapter, *first*, we explained the workflow and all the technologies involved in streaming a video over the Internet. We provided a birds-eye-view of how these technologies interact with each other to achieve a high quality of experience for viewers. *Second*, we provided details on each of those technologies and challenges the video streaming researchers and practitioners are encountering in each of those areas. *Third*, we reviewed the ways various cloud services can be leveraged to cope with the demands of video streaming services.

In the next chapter, our focus is directed toward video streaming using cloud services. Particularly, we explore the computational complexity of transcoding operation and conduct simulations to measure the transcoding performance on different types of VMs in the cloud. Then, we analyze the execution time of different transcoding processes on heterogenous VMs. Furthermore, we carry out analysis on video content type, GOP size, frames, and transcoding time. We also come up with a model to trade-off between performance and cost of transcoding tasks in the cloud.

CHAPTER 3: Analysis of Cloud-Based Video Transcoding

3.1. Introduction

To minimize the video streaming delay for such diverse viewers, Streaming Service Providers (SSPs) commonly *pre-transcode* videos, *i.e.,* they store several versions of the same video [95]. Given the volume of videos that needs to be transcoded and stored, this approach requires massive storage and processing resources. Provisioning and upgrading built-in infrastructures to meet these demands is cost-prohibitive and distracts SSPs from their mainstream business, which is producing video content and focusing on viewers' satisfaction. Therefore, SSPs have become extensively reliant on cloud providers to provide their services [10].

Cloud providers offer an abundance of reliable computational and storage services to SSPs. Making use of cloud services, however, poses new challenges to SSPs. One main challenge is to minimize the incurred cost for using cloud services while maintaining QoS (in terms of uninterrupted streaming experience) for their viewers. To overcome this challenge, several research works have been undertaken in estimating video transcoding time [17, 9], video segmentation models [18, 10], scheduling [19, 17], and resource provisioning methods [10, 20]. However, these studies generally focus on the elasticity aspect of cloud VMs. That is, how VMs can be allocated or deallocated to maximize the QoS satisfaction and minimize the incurred cost of SSPs.

Cloud providers offer a wide variety of VM types (*i.e.,* heterogeneous VMs) with diverse prices. For instance, Amazon EC2 offers VM types, such as `General-Purpose`, `CPU-Optimized`, and `GPU` that have different architectural characteristics and remarkably diverse costs. In such a heterogeneous environment, different transcoding operations (also

termed *transcoding task*) can potentially have various transcoding times (*i.e.,* execution times) on the heterogeneous VMs. The reason for the various execution times is the *affinity* of different transcoding operations with different VM types. For instance, particular transcoding tasks can be CPU-intensive and execute faster on `CPU-Optimized` VMs whereas some other transcoding tasks can be memory-intensive. More importantly, some transcoding tasks can have similar transcoding times on heterogeneous VMs while their incurred costs vary significantly.

Understanding the affinity between transcoding tasks and heterogeneous VMs is critical in scheduling and VM provisioning decisions of SSPs. Such decisions should rely on accurate performance information of transcoding tasks and their incurred costs on heterogeneous VMs. Hence, a deep understanding and analysis of the affinity of transcoding tasks with heterogeneous cloud VMs are required. Currently, there is no study of this kind available.

Expected Time to Compute (ETC) [119, 120] and *Estimated Computation Speed* (ECS) [121, 122] matrices are commonly used to model and explain the affinity of tasks on heterogeneous machines. However, the definition of both ETC and ECS considers only the execution time as the performance metric and ignores the cost difference across different VM types. The question that arises how we can have a model that captures both the execution time and cost differences of heterogeneous cloud VMs? Answering this question can be useful for resource (VM) provisioning methods to allocate appropriate type of VMs for incoming transcoding tasks.

In summary, the **research questions** we address in this research are:

- How can we recognize the affinity of different transcoding tasks with heterogeneous

cloud VMs?

- How can we model the trade-off between performance and cost of heterogeneous VMs for different transcoding tasks?

To answer the first question, we identify if video transcoding tasks have different affinities with heterogeneous cloud VMs. Then, we need to recognize how the affinity is defined. That is, finding an appropriate factor in video transcoding tasks to determine the affinity of transcoding tasks on heterogeneous VMs. In particular, we investigate two factors that can determine the affinity of transcoding tasks with heterogeneous VMs, namely type of transcoding operation and video content type.

For that purpose, we analyze the affinity of transcoding tasks on heterogeneous cloud VMs when the tasks are categorized based on the type of their transcoding operation and when they are categorized based on their content types. The result of these analyses shows that video content type is a stronger factor to determine the affinity of video transcoding tasks on heterogeneous VMs rather than the type of transcoding operation. However, it is difficult to categorize video transcoding tasks based on their content type because the content type is not known before the execution of the tasks. Hence, in the next step, we find factors that indicate the video content type and, as such, can be used for categorizing video transcoding tasks.

To answer the second research question, we present a model to quantify the suitability of heterogeneous VMs for a given transcoding task. The model encompasses both the execution time of the task on a VM type and the incurred cost of using it.

In summary, the key **contributions** of this chapter are:

- Analyzing the performance of different transcoding operations on heterogeneous

cloud VMs.

- Analyzing the performance of video content types on the on heterogeneous cloud VMs.

- Determining influential factors on the execution time of the transcoding operation.

- Providing a model to capture (and quantify) the cost and performance trade-off of heterogeneous VMs for video transcoding tasks.

3.2. Heterogeneous Computing in Cloud

Cloud service providers offer heterogeneous computational services (VMs) to satisfy various types and levels of computational requirements of their clients. Heterogeneity of these VMs is based on both underlying hardware characteristics and their hourly cost. Such heterogeneity enables cloud users to build a Heterogeneous Computing (HC) environment (*i.e.,* a cluster of heterogeneous VMs) to efficiently process high-performance computations in the cloud environment.

HC environments are categorized as *consistent* and *inconsistent* [121] heterogeneous environments. The former refers to an HC environments in which some machines (VMs) are consistently faster than others whereas the latter explains an environment in which tasks have diverse execution times on heterogeneous machines. For instance, machine A may be faster than machine B for task 1 but slower than other machines for task 2 [123]. We also say that machine A has a higher affinity with task 1. In fact, cloud providers offer several categories of VMs that are inconsistently heterogeneous. Nevertheless, there is a consistent heterogeneity within VMs in each one of those

categories. In this study, our goal is to study the affinity of different transcoding tasks on heterogeneous VMs. Thus, we consider a cloud as an inconsistently HC environments.

In the case of Amazon EC2 cloud[a], six categories of VM types are offered that are described below:

- **General-Purpose VMs:** This VM type has a fair amount of CPU, memory, and networks for many applications, such as web servers and small- or mid-size database servers. General-purpose VMs is the least expensive one and have lower computing power in comparison with other VM types. To process a large set of tasks, either many or few of these VMs should be allocated for a long time [99].

- **CPU-Optimized VMs:** This VM type offers a higher processing power in comparison with other VM types, which makes them ideal for compute-intensive tasks. They are currently mostly applied to high-traffic web application servers, batch processing, video encoding, and high-performance computing applications (*e.g.,* genome analysis and high-energy physics) [100].

- **Memory-Optimized VMs:** Memory-Optimized VM type is designed for processing tasks with large memory demand. The VM type has the lowest cost per GB of memory (RAM) compared to other types. Applications such as high-performance databases, distributed cache, and memory analytics [124] usually demand Memory-Optimized VMs.

- **GPU-Optimized VMs:** The GPU-Optimized VMs are applied for compute-intensive tasks (*i.e.,* tasks that involve huge mathematical operations).

[a] https://aws.amazon.com/ec2/

Many large-scale simulations, such as computational chemistry, rendering, and financial analysis are conducted on GPU-Optimized VMs [125].

- **Storage-Optimized VMs:** Storage-Optimized VMs are utilized in cases where low storage cost and high data density is necessary. This VM type is designed for large (big) data requirements such as Hadoop clusters and data warehousing applications [126].

- **Dense-Storage VMs:** These VMs provide large Hard Disk Drive (HDD) storage with low price per disk throughput performance. This VM type benefits applications such as data warehousing, and Hadoop-based applications [127].

We perform this research by using VM types offered by the Amazon cloud provider. The reason we chose Amazon is that it provides reliable and heterogeneous VM types. Also, currently, Amazon is the mainstream cloud provider and many video SSPs utilize its services [14]. However, we would like to note that the analysis provided in this work is general and can be applied to any HC environment.

3.3. Performance Analysis of Video Transcoding Operations on Heterogeneous Cloud VMs

3.3.1. Overview

To keep the generality and to avoid limiting the research to the details of VM types offered by Amazon EC2, we select one VM type from different VM categories in Amazon EC2 that represents the characteristics of that category (see Section 3.2).

In particular, for the General-Purpose, CPU-Optimized, Memory-Optimized, and GPU VM types we choose `m4.large`, `c4.xlarge`, `r3.xlarge`, and `g2.2xlarge`, respectively. We did not consider any of the Storage-Optimized and Dense-Storage VM

types in our evaluations as we observed that IO and storage are not influential factors for video transcoding tasks. The characteristics and the cost of the chosen VM types are illustrated in Table 3.1. In this table, vCPU represents virtual CPU. Amazon uses what it calls "EC2 Compute Units" or ECUs, as a measure of virtual CPU power. It defines one ECU as the equivalent of a 2007 Intel Xeon or AMD Opteron CPU running at 1 GHz to 1.2 GHz clock rate. More details about the characteristics of the VM types can be found at Amazon EC2 website[b].

Table 3.1. Cost of heterogeneous VMs in Amazon EC2 cloud.

VM Type	General (`m4.large`)	CPU Opt. (`c4.xlarge`)	Mem. Opt. (`r3.xlarge`)	GPU (`g2.xlarge`)
vCPU	2	4	4	8
Memory (GB)	8	7.5	30.5	15
Hourly Cost ($)	0.15	0.20	0.33	0.65

3.3.2. Analyzing the Execution Time of Different Video Transcoding Operations

The first question we need to answer is whether or not a certain transcoding operation has more affinity to a particular type of VM in the cloud?

To answer this question, we compared the transcoding time (execution time) of various transcoding operations using different VM types. In particular, we measured the transcoding time of the first nine GOPs in all videos in the benchmark on different VM types and reported and the mean of their transcoding times. The reason we choose nine GOPs is that the shortest video exists in the benchmark has nine GOPs.

Figure 3.1 shows the transcoding time of different transcoding operations on

[b] https://aws.amazon.com/ec2/instance-types/

heterogeneous VMs. We can observe that the execution times of different transcoding operations are not the same, however, regardless of the VM type, they follow the same pattern.

Sub-figures 3.1a, 3.1b, 3.1c, and 3.1d demonstrate that although the execution time of each transcoding operation varies on different VM instances, in general, transcoding time has the same pattern across `General`, `CPU Opt.`, `Mem. Opt.` and `GPU` VM types.

The results confirm that, regardless of the VM type utilized, converting video codec always takes more time than other transcoding operations. This is because changing codec implies the whole original encoded video be decoded and then re-encoded with new one codec. These conversions make the transcoding time longer.

We also observe that changing resolution has the least transcoding time regardless of the VM type. The reason is that the transcoding is achieved by utilizing filtering and subsampling [128, 129] which works directly in the compressed domain and avoids the computationally expensive steps of converting to the pixel domain. Therefore, it takes less time than other transcoding operations.

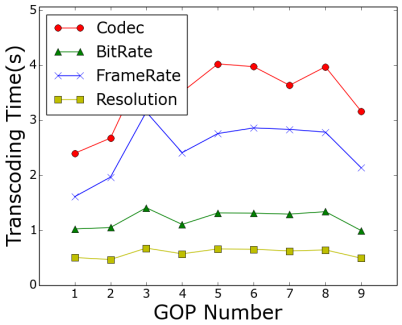
To analyze the transcoding time, we utilized a set of benchmark videos. The benchmarking videos are publicly available for reproducibility purposes[c]. Videos in the benchmark diverse both in terms of the content types and length. The benchmark includes a combination of slow, fast, and mixed motion video content types. The length of the videos in the benchmark varies in the range of [10, 600] Seconds.

We used `FFmpeg`[d], which is an open source utility, to transcode the videos. For each one of the benchmarking videos, four different transcoding operations, namely codec

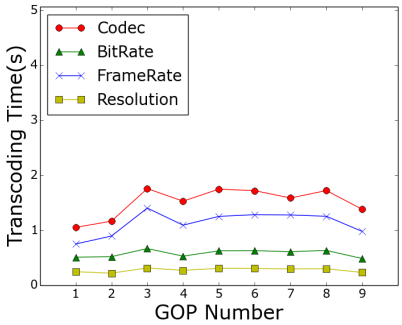
[c]The videos can be downloaded from: https://goo.gl/TE5iJ5

[d]https://ffmpeg.org

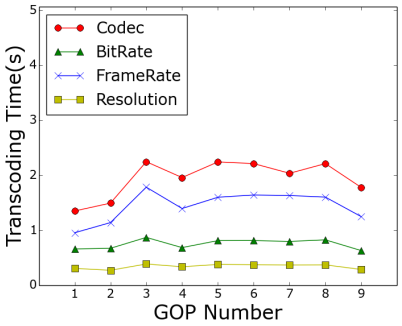
Figure 3.1. Mean transcoding time (in Seconds) on GOPs of benchmark videos on distinct VM types. (a) mean transcoding time of different transcoding operations on `General` VM. (b), (c), and (d) show mean transcoding time of different transcoding operations using `CPU Opt.`, `Mem. Opt.`, and `GPU`, respectively.



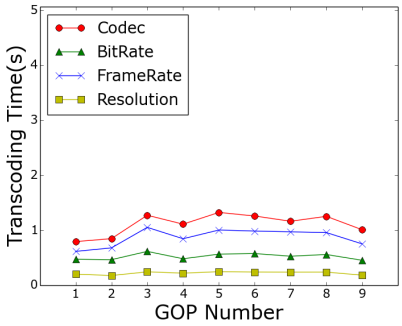
(a) Time taken by different transcoding operations on `General` VMs



(b) Time taken by different transcoding operations on `CPU Opt.` VMs



(c) Time taken by different transcoding operations on `Mem. Opt.` VMs



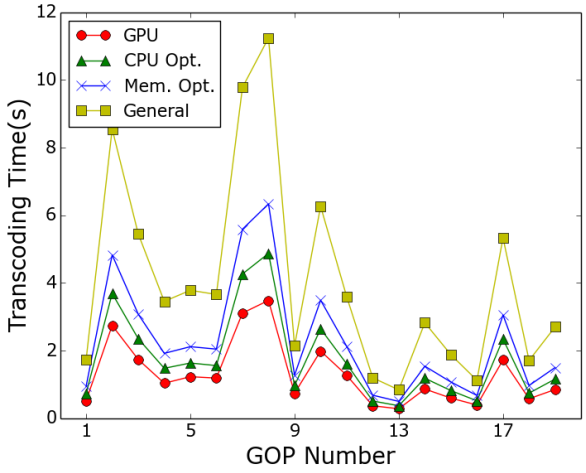
(d) Time taken by different transcoding operations on `GPU` VMs

conversion, resolution reduction, bit rate adjustment, and frame rate reduction were carried out on heterogeneous VMs.

Each transcoding operation has been repeated for 30 times on each video to remove any randomness (*e.g.,* due to VM malfunctioning or other temporal issues)[e]. The mean transcoding time on each VM for a given GOP is considered for comparison and analysis of this chapter.

3.3.3. Analyzing the Affinity of Video Transcoding Operations with Heterogeneous Cloud VMs

Figure 3.2. Execution time of codec transcoding for a video in the benchmark on different VM types.



As mentioned in Section 3.2., cloud providers offer VMs that are heterogeneous both in terms of performance and cost. An important question for video stream providers to reduce their cost and improve their Quality of Service (QoS) is: what is the affinity of video transcoding operations with heterogeneous cloud VMs?

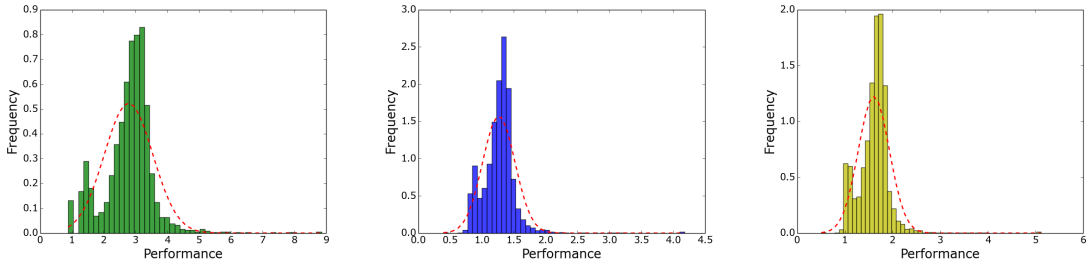
As we noticed in Section 3.3.2., although execution times of various video

[e]The generated workload traces are available publicly from: https://goo.gl/B6T5aj

transcoding operations were different, codec transcoding constantly has the highest execution time and changing spatial resolution generally has the lowest execution time. Considering the pattern in transcoding execution time, to study the affinity of video transcoding on heterogeneous VMs, we only need to consider one transcoding operation (*e.g.,* codec transcoding) on heterogeneous VMs. Hence, we measure the codec transcoding time of benchmark videos on heterogeneous VM types.

Figure 3.2 expresses the analysis for one video[f] in the benchmark. Graph of the same evaluation is illustrated in APPENDIX A for other videos of the benchmark. In this figure, we can observe that, in general, `GPU` VM provides a better execution time in comparison with other VM types. This is because transcoding operations include substantial mathematical operations and `GPU` VM types are well suited for such kind of operations. `General` VM provides the lowest performance as it includes less powerful processing units (see Table 3.1).

Figure 3.3. Performance ratio of transcoding on different VM types with respect to `GPU` VM. Horizontal axis shows the performance ratio and the vertical axis shows the frequency of the performance ratio for all GOPs of videos in the benchmark. (a) Shows the performance ratio of `General` VM with respect to `GPU`. (b) Shows the performance ratio of `CPU Opt.` VM with respect to `GPU` VM. (c) Shows the performance of `Mem. Opt.` VM with respect to `GPU` VM.



(a) Performance ratio of `General` to `GPU`. (b) Performance ratio of `CPU-Opt.` to `GPU`. (c) Performance ratio of `Mem-Opt.` to `GPU`.

[f]This is `big_buck_bunny_720p_h264_02tolibx264` video in the benchmark.

More importantly, in Figure 3.2, we observe that the transcoding times of different GOPs are significantly varying on the four VM types. For some GOPs, the `GPU` VM remarkably outperforms other VMs (*e.g.,* GOP 6, 7, and 8) whereas for some other GOPs (*e.g.,* GOP 9, 12, and 13) the difference in transcoding times is negligible.

To better understand the performance variations in transcoding different GOPs, we compared the performance of these four VM types for all videos in the benchmark in detail. Although `GPU` takes the least time to perform a transcoding operation, we are interested to know the significance of the outperformance of the `GPU` across different GOPs. Thus, we define *performance ratio* for GOP i as its transcoding time on a VM type to the transcoding time on `GPU`. The result of this analysis is shown in Figure 3.3. In all sub-figures of Figure 3.3, the horizontal axis shows the performance ratio and the vertical access shows the frequency of that ratio across all GOPs in a video. According to Figure 3.3, we observe that:

- In Sub-figure 3.3a, performance ratio of `General` VM lies within the range 2.781 ± 1.524.
- In Sub-figure 3.3b, performance ratio of `CPU Opt.` VM lies within the range 1.263 ± 0.508.
- In Sub-figure 3.3c, performance ratio of `Mem. Opt.` VM lies within the range 1.608 ± 0.652.

We also measured the percentage of GOPs transcoded on VMs other than `GPU` with performance ratio < 1.0. That is, the percentage of GOPs that their transcoding time is less than the transcoding time on the `GPU`. The results are shown in Table 3.2.

Table 3.2. Performance ratio of different VM types with respect to GPU VM, for all GOPs of the videos in the benchmark. Each entry shows the percentage of GOPs with performance ratio < 1.0.

	Codec	Frame Rate	Bit Rate	Resolution
General	0%	2.4%	2.7%	2.4%
CPU Opt.	2.2 %	**24.8**%	**28.0**%	3.9%
Mem. Opt.	0.6%	2.8%	4.2%	2.5%

Table 3.3. Performance ratio of different VM types with respect to GPU VM, for all GOPs of the videos in the benchmark. Each entry shows the percentage of GOPs with performance ratio ≤ 1.2.

	Codec	Frame Rate	Bit Rate	Resolution
General	0%	2.72%	2.87%	2.72%
CPU Opt.	12.28%	**33.33**%	**63.93**%	22.28%
Mem. Opt.	1.36%	23.78%	23.63%	3.49%

Entries of this table express the percentage of GOPs that have transcoding time strictly lower than the GPU for different transcoding operations. In addition, to see the percentage of transcoding tasks that have close execution time to the GPU, in Table 3.3, the percentage of tasks that have performance ratio lower than 1.2 are reported.

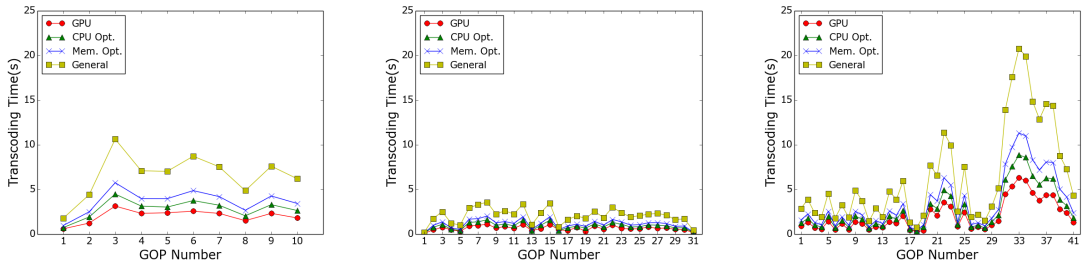
We can summarize our observations in this part as follows:

- We observe that in cases that transcoding time of other VM types are lower than GPU, the transcoding time differences are low (less than 0.24 seconds).
- In all cases that other VM types outperform the GPU VM, the transcoding time on the GPU was low (less than 2.1 seconds). That is, when the GPU takes a low time to transcode a GOP, other VM types may outperform it.

- According to Figure 3.3, none of the transcoding types need extensive memory space (*i.e.,* transcoding is not a memory intensive operation). Therefore, video stream providers would not benefit from instantiating memory-optimized VM types.

3.3.4. Analyzing the Impact of Video Content Type on Transcoding Time

Figure 3.4. Transcoding time (in Seconds) of video streams on different cloud VMs for various video content types. (a), (b), and (c) demonstrate the transcoding time obtained from different VM types when applied on slow motion, fast motion and mixed motion video content types, respectively.



(a) Transcoding time of slow motion video.

(b) Transcoding time of fast motion video.

(c) Transcoding time of mixed motion video.

As we observed in the previous section, the transcoding time of a GOP can vary significantly on different VM types (see Figure 3.2). For instance, transcoding time difference between `GPU` and `CPU Opt.` VM types for GOP 8 in Figure 3.2 is $\simeq$ 7 seconds while the difference for GOP 13 is less than a half second. What is this performance difference attributed to? Answering this question enables us to allocate the appropriate VM types depending on the GOP type, hence, reducing the transcoding time and its incurred cost.

Our investigation revealed that the reason for the transcoding time variations is the content type of the GOPs. To further investigate the impact of video content type on the transcoding performance, we performed codec transcoding on each video content type

on different VM types. Results of the investigation are reported in Figure 3.4.

Figure 3.4a shows that the transcoding time of the slow motion videos is distinct from each other across different VM types. In particular, `GPU` and `General` VM types, respectively, provide the best and worst performance for this type of video content.

In contrast, Figure 3.4b shows that the outperformance of `GPU` VM is not statistically and practically significant when transcoding fast motion videos. Although `GPU` still provides a slightly faster transcoding time than other VM types, the difference is negligible. For some GOPs (*e.g.,* 4, 5, 13, 16, and 31) the transcoding time on GPU is almost the same as other VM types.

To confirm this finding, we performed the transcoding operation on a mixed motion video and the result is depicted in Figure 3.4c. As we can see in this sub-figure, `GPU` outperforms others VMs significantly for some GOPs (*e.g.,* . GOP 30 to 37) while provides almost same transcoding time for other GOPs. We noticed that the difference in transcoding time is remarkable for GOPs of the video that contains slow motion content and it is negligible for fast motion GOPs.

The reason for the performance variations on different video content types is that, in fast motion videos, due to the high frequency of changing scenes, the number of frames in a GOP and, therefore, the GOP size is small. In contrast, slow-motion GOPs include more frames, and they are larger. When we transcode a large number of small size GOPs (*i.e.,* the case for fast motion videos), there is little computation to be performed for each GOP and the performance of the VM is dominated by the overhead of switching between different GOPs. On the contrary, when in transcoding slow motion videos we deal with few number of GOPs that are large (*i.e.,* they are compute intensive). Transcoding such

videos can take advantage of compute-heavy (*e.g.,* `GPU`) VMs.

In the next section, we will further investigate the impact of GOP size and number of frames in a GOP for video transcoding.

3.3.5. Analyzing the Impact of GOP Size and Number of Frames on Transcoding Time

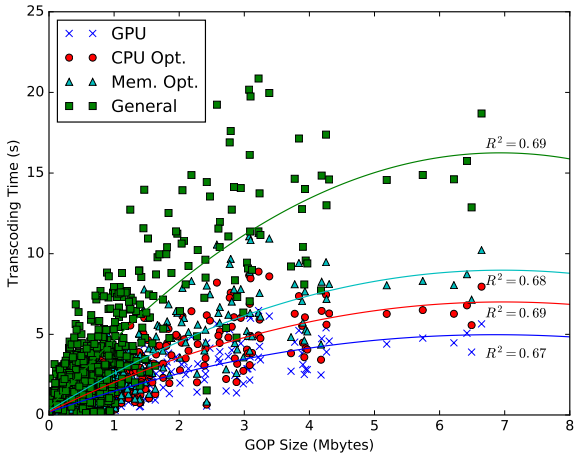
In Section 3.3.4., we concluded that the transcoding time of GOPs varies significantly on different VMs depending on the video content types. However, automatic categorization of GOPs based on their video content type is a difficult task. We need an intuitive factor to categorize GOPs on different VM types (*i.e.,* to identify the affinity of GOPs with heterogeneous VMs). In this section, we investigate further the factors that influence GOP transcoding time on different VM types.

As we noticed in Section 3.3.4., a GOP with slow motion content type benefits more from a computationally powerful VM. Such a GOP has a large size and includes many frames. Therefore, we need to analyze the impact of GOP *size* and *number of frames* on the transcoding time of each GOP on different VM types.

We use a regression analysis to study the impact of GOP size and number of frames on the GOP transcoding time. We consider the transcoding time of GOPs in all benchmark videos of the benchmark that is a mixture of slow, fast, and mixed motion video contents. Due to a large amount of data, the second-degree regression is used for the analysis. The horizontal axis shows The GOP size (in MB) and number of frames for all GOPs in Figures 3.5 and 3.6, respectively. The vertical axis in both figures shows the transcoding times of GOPs (in seconds).

In both figures, we observe that, regardless of the VM type, transcoding times increase by increasing both the GOP size and the GOP number of frames. Table 3.4

Figure 3.5. Second degree regression to study the influence of GOP size on the transcoding time.



demonstrates the coefficient of determination (R^2) for the regression analyses. As we can see in this table, both GOP size and number of frames show the high confidence of relationship to transcoding time, while the number of frames in a GOP shows a higher R^2 value for all VM types. Therefore, many frames provide a stronger regression with transcoding time.

In Figure 3.6, we also observe that when the number of frames in a GOP is small, the performance of `GPU` is very close to other VM types whereas for a larger number of frames, the performance gap between GPU and others VM types rises. This implies that GOPs with few number of frames are better to be assigned to cost-efficient VM types whereas GOPs with a large number of frames can benefit from computationally powerful VM types.

Figure 3.6. Second degree regression to study the influence of number of frames in a GOP on the transcoding time.

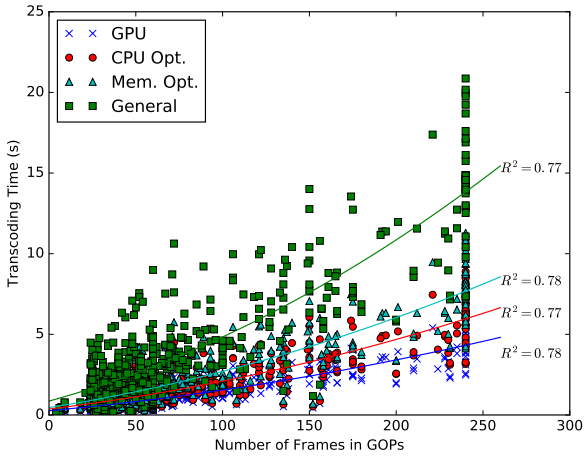


Table 3.4. Coefficient of determination (R^2) for regression analysis of GOP size and number of frames with transcoding time.

	General (`m4.large`)	CPU Opt. (`c4.xlarge`)	Mem. Opt. (`r3.xlarge`)	GPU (`g2.xlarge`)
GOP Size	69.68%	68.76%	69.11%	67.56%
Number of Frames	77.15%	78.16%	77.31%	78.17%

3.4. Modeling performance Cost Trade-Off of transcoding Tasks on Heterogeneous Cloud VMs

VM types offered by cloud providers are heterogeneous both in terms of performance and cost [130]. Hence, allocating VMs that are cost- and performance-efficient for transcoding tasks is challenging.

As we discussed in Section 3.3.3., computationally-powerful VMs do not always provide the best performance for transcoding tasks. This is particularly important when we consider the significant cost difference between the VM types. We also discussed that the transcoding time has a correlation with the GOP size and number frames in GOPs. In

particular, when the GOP size or number of frame is small, the performance difference of heterogeneous VMs is negligible.

Alternatively, the performance difference of using heterogeneous VMs to transcode large size GOPs is significant. Thus, it may be worthwhile to allocate a powerful and costly VM to transcode such GOPs.

To cope with the appropriate VM type allocation challenge, we require a construct to identify the appropriateness of various VM types for different GOPs. Such a construct can be helpful in allocation and mapping (*i.e.,* scheduling) of GOPs to the appropriate VMs for transcoding.

In this section, we present a construct termed GOP *Suitability Matrix* that maintains the suitability value of each VM type for each GOP task in a video stream. Such a matrix can be used by video stream providers to allocate VMs that offer the best performance and cost trade-off for video transcoding.

Recall from Table 3.1 that `GPU` VM type, in general, provides the best performance while having the highest cost. Also, `General` VM type provides the lowest transcoding performance and is the least expensive one when compared to other VMs.

We define *performance gap*, denoted Δ_i, as the performance difference VM type i and `GPU`. For a given GOP, a large value of Δ_i indicates that VM type i remarkably performs worse than `GPU`, hence, `GPU` should be assigned a higher suitability value than VM i.

Determination of the trade-off between performance and cost of utilizing heterogeneous VMs, in the first place, depends on the business policy of the streaming service provider (here, we call it *user*). That is, a user should determine how important is

the performance, denoted p, and incurred the cost, denoted c, for the system. As these parameters complement each other (*i.e.,* $p + c = 1$), the user only needs to provide one of these parameters. For instance, a user can provide $p = 0.6$ (that implies $c = 0.4$) to indicate a higher performance preference.

We define *performance threshold gap*, denoted Δ_{th}, as the threshold of the performance gap between `GPU` and other VM types. The value of Δ_{th} is determined based on the user preference of p and c. As user cost and performance preferences are not crisp values, we can model them based on fuzzy membership functions [131]. As shown in Figure 3.7, we define two membership functions for the cost and performance preferences. According to this figure, the membership value of one preference (*e.g.,* performance) decreases when the other preference (*e.g.,* cost) increases.

Value of the user's performance (or cost) preference is considered as the membership value of the fuzzy membership function (vertical axis in Figure 3.7) and is used to obtain the performance threshold gap (horizontal axis in Figure 3.7). More specifically, by using the performance preference (p), we can obtain Δ_{th} based on Equation (3.1)[g].

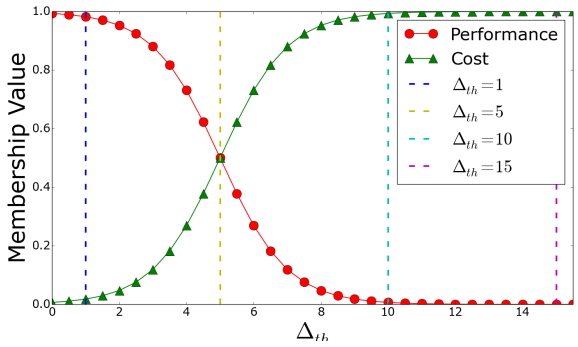
$$\Delta_{th} = \frac{\ln \frac{p}{1-p}}{\alpha} + \beta \tag{3.1}$$

Where α is the inflection point in the membership function and β is the slope at α. In Figure 3.7, we experimentally obtained the values of α and β equal to 5 and 0.7, respectively.

Based on the value of Δ_{th}, we can determine the trade-off between performance

[g]Similarly, the value of Δ_{th} can be obtained from the cost preference value: $\Delta_{th} = \frac{\ln \frac{c}{1-c}}{\alpha} - \beta$

Figure 3.7. Membership functions for performance and cost preferences. The user-provided values for the cost or performance are considered as the membership value (vertical axis). Then, the corresponding value on the horizontal axis is considered as Δ_{th}



and cost for transcoding a given GOP. For that purpose, we define *weight* of the VM type i, denoted W_i, to transcode a given GOP based on Equation 3.2 that encompasses both the performance and cost factors.

The first part, in Equation 3.2, considers the performance factor and calculates the difference of performance gap from Δ_{th}. Performance gaps greater than the threshold (Δ_{th}) cause a low (negative) weight value which implies higher Suitability for performance-oriented VM types. In this part, the denominator determines the sum of performance gaps for all N VM types. The second part, in Equation 3.2, considers the cost factor. This part functions based on the cost of transcoding a given GOP on VM type i, denoted φ_i. The cost of transcoding a GOP on VM i is obtained from the transcoding time of the GOP on VM i and the hourly cost of VM type i in the cloud. This part of the equation favors VM types that incur a lower cost for transcoding a given GOP.

$$W_i = \frac{\Delta_{th} - \Delta_i}{\sum_{n=1}^{N} \Delta_n} \cdot \left(1 - \frac{\varphi_i}{\sum_{n=1}^{N} \varphi_n}\right) \tag{3.2}$$

To normalize the value of W_i and determine the final suitability values, denoted

S_i, between [0, 1] we use Equation (3.3) as follows:

$$S_i = \frac{W_i - \max_i(W_i)}{\max_i(W_i) - \min_i(W_i)} \tag{3.3}$$

where max_i and min_i are the largest and smallest values among W_is, respectively. In transcoding a video stream, each GOP has different suitability values on different VM types. These suitability values construct a *Suitability Matrix* for each video stream.

To have a better understanding of the Suitability Matrix construct, we compare four suitability matrices with different performance and cost preference values provided by the user.

Table 3.5a shows the Suitability Matrix for a given video when the user has a performance-oriented preference—$p = 0.98$. As we can see, in this case, the Suitability value of `GPU` and `CPU Opt.` VMs is higher than the other VM types. We observe that the Suitability values for (`General`) VM are mostly 0.

When user's performance preference drops to 0.5 (and cost raises to 0.5), as demonstrated in Table 3.5b, the Suitability value of `GPU` decreases while the `General` VM gets higher Suitability values. By further decreasing the user performance preference and increasing the cost preference, the Suitability value of the `GPU` drops to almost 0 while the Suitability values of cost-efficient VMs (`General`) are increased (see Tables 3.5c and 3.5d). It is noteworthy that `CPU Opt.` VM type mostly maintains a high Suitability value regardless of Δ_{th} value. This is because the `CPU Opt.` VMs has a high performance and its cost is relatively low.

3.5. Summary

In this chapter, we provided a detailed study and analysis of different transcoding operations on heterogeneous VMs in terms of execution time, performance, and cost of using cloud resources. As a result, we modeled a trade-off between performance and the cost of using cloud resources. More importantly, we came up with a significant correlation between transcoding time and GOP size in a video stream.

In the next chapter, we employ this correlation in synthesizing video streams repositories. And thus we propose methods to trade-off between pre-transcoding, re-transcoding and partial pre-transcoding of video streams in repositories, to reduce the cost of using cloud resources.

Table 3.5. Suitability Matrices for different values of performance and cost preferences. Tables (a) to (d), show that as the performance preference p decrease (and cost-preference c increases), the value of Δ_{th} grows. Accordingly, the maximum Suitability value changes from `GPU` (performance-oriented VM) in Table 3.5a to `General` type (cost-oriented VM) in Table 3.5d.

VM Type	`General`	`CPU Opt.`	`Mem. Opt.`	`GPU`
GOP_1	**0.00**	1.00	0.78	**0.98**
GOP_2	**0.00**	1.00	0.68	**0.26**
GOP_3	**0.00**	1.00	0.67	**0.30**
GOP_4	**0.00**	1.00	0.61	**0.01**
GOP_5	**0.00**	1.00	0.71	**0.60**
GOP_6	**0.00**	1.00	0.80	**0.89**
GOP_7	**0.00**	0.91	0.74	**1.00**
GOP_8	**0.00**	0.88	0.72	**1.00**
GOP_9	**0.00**	0.87	0.72	**1.00**
GOP_{10}	**0.00**	0.86	0.71	**1.00**
⋮	⋮	⋮	⋮	⋮

(a) Suitability Matrix, when $p = 0.98\%$, $c = 0.2\%$, and $\Delta_{th} = 1$

VM Type	`General`	`CPU Opt.`	`Mem. Opt.`	`GPU`
GOP_1	0.00	1.00	0.63	0.03
GOP_2	0.69	1.00	0.78	0.00
GOP_3	0.67	1.00	0.78	0.00
GOP_4	0.75	1.00	0.78	0.00
GOP_5	0.57	1.00	0.74	0.00
GOP_6	0.26	1.00	0.71	0.00
GOP_7	0.00	1.00	0.72	0.54
GOP_8	0.00	1.00	0.75	0.70
GOP_9	0.00	1.00	0.77	0.77
GOP_{10}	0.00	1.00	0.77	0.77
⋮	⋮	⋮	⋮	⋮

(b) Suitability Matrix, when $p = 0.5\%$, $c = 0.5\%$, and $\Delta_{th} = 5$

VM Type	`General`	`CPU Opt.`	`Mem. Opt.`	`GPU`
GOP_1	0.48	1.00	0.73	0.00
GOP_2	0.79	1.00	0.80	0.00
GOP_3	0.79	1.00	0.80	0.00
GOP_4	0.83	1.00	0.80	0.00
GOP_5	0.74	1.00	0.78	0.00
GOP_6	0.59	1.00	0.77	0.00
GOP_7	0.06	1.00	0.65	0.00
GOP_8	0.00	1.00	0.68	0.20
GOP_9	0.00	1.00	0.71	0.35
GOP_{10}	0.00	1.00	0.70	0.32
⋮	⋮	⋮	⋮	⋮

(c) Suitability Matrix, when $p = 0.5\%$, $c = 0.5\%$, and $\Delta_{th} = 10$

VM Type	`General`	`CPU Opt.`	`Mem. Opt.`	`GPU`
GOP_1	**0.63**	1.00	0.76	**0.00**
GOP_2	**0.82**	1.00	0.81	**0.00**
GOP_3	**0.83**	1.00	0.81	**0.00**
GOP_4	**0.85**	1.00	0.81	**0.00**
GOP_5	**0.79**	1.00	0.80	**0.00**
GOP_6	**0.69**	1.00	0.79	**0.00**
GOP_7	**0.37**	1.00	0.72	**0.00**
GOP_8	**0.19**	1.00	0.69	**0.00**
GOP_9	**0.04**	1.00	0.66	**0.00**
GOP_{10}	**0.08**	1.00	0.67	**0.00**
⋮	⋮	⋮	⋮	⋮

(d) Suitability Matrix, when $p = 0.01\%$, $c = 0.99\%$, and $\Delta_{th} = 15$

CHAPTER 4: Video Streaming Repository Management

4.1. Introduction

To make the video streams readily available for viewers, video streams providers commonly carry out the transcoding operation in an offline manner. That is, they transcode and store (termed *pre-transcode*) multiple formats of the same video stream to satisfy the requirements of viewers with heterogeneous display devices. In practice, video stream providers such as Netflix have to pre-transcode 40 to 50 formats of a single video stream [7] and store them in their repositories. To overcome the storage and computational demands of transcoding, video stream providers extensively utilize cloud services [8]. Making use of cloud services imposes a significant cost overhead to video stream providers [9, 10], hence, they are actively seeking solutions to reduce their incurred cost of using cloud services [132].

To reduce the incurred cost, video stream providers need to be aware of access patterns to their video streams. Recent studies show that accessing video streams in a repository follows a long-tail distribution [11]. That is, there are few video streams that are accessed very frequently (known as *hot* videos) while there is a huge portion of video streams in the repository that is rarely accessed. Several research works have been undertaken (*e.g.,* [10, 9]) to alleviate the cost overhead of pre-transcoding by transcoding rarely-accessed videos in an on-demand (*i.e.,* lazy) manner. In this manner, one or few formats of a video is pre-transcoded, and transcoding is performed on-the-fly upon request to access a version of a video that is not already pre-transcoded. This approach has become feasible with the enormous computational capacity clouds offer and is known as *re-transcoding*.

Re-transcoding induces a computational cost that is more expensive than the storage cost [12]. Therefore, the re-transcoding approach would be beneficial for video stream providers, only if it is applied on rarely accessed videos (also termed *non-hot* or *cold* video streams). To decide whether or not to pre-transcode or re-transcode a video stream, we need to know if the video stream is hot or not. However, the question that arises is *what is a hot video stream?* Currently, there is no formal way to determine if a video stream is hot or, more precisely, there is no formal way to measure the hotness of a video stream. Therefore, the research problem in this research is *how can we quantify the hotness of a given video in a repository?*

Once we answer this question, we can decide to pre-transcode hot video streams and re-transcode non-hot ones. Besides, we can *partially pre-transcode* video streams whose their hotness measure is between hot and non-hot. The challenge, however, is how to achieve partial pre-transcoding? That is, which parts of a video stream should be re-transcoded and which parts should be pre-transcoded? To address this challenge, in this chapter, we propose *video stream repository management methods* (also termed *repository management*) to perform pre-transcoding based on the hotness of the video streams. We also propose a management method of video stream repository that operates at a finer granularity level and can accurately specify the parts of the video streams to be pre-transcoded.

The repository management methods that we propose in this chapter must be applied to all video streams in a repository. However, in a large video stream repository, the repository management methods impose a large overhead time to execute. We require a way to alleviate the overhead time of executing the repository management methods. To

address this challenge, in this research, we propose a method that partitions hot, non-hot, and partially hot video streams into separate clusters. The clustering can help to proactively apply repository management methods only on one or few clusters, and hence, reduce the execution time overhead of repository management methods.

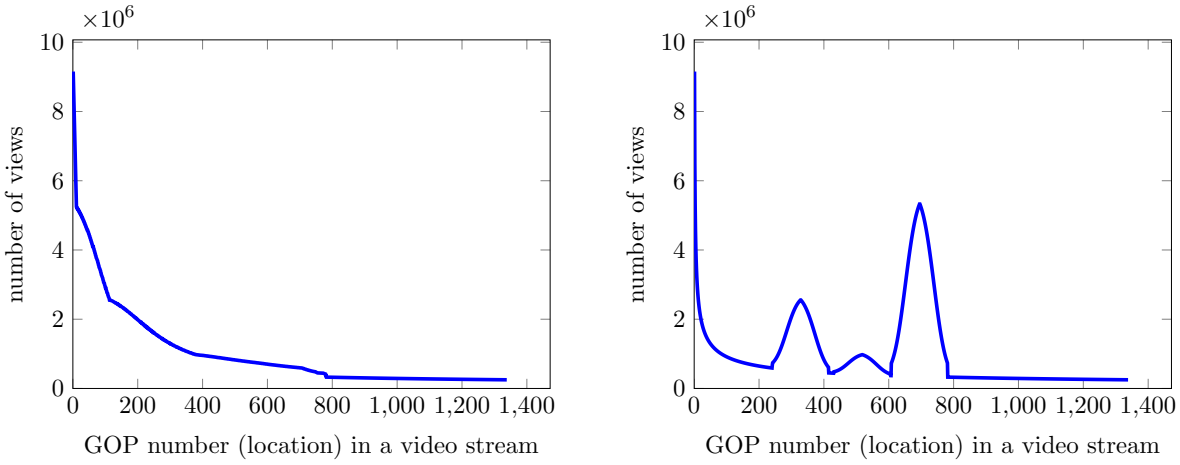
In summary, the contributions of this chapter are as follows:

- Providing a formal definition for hot videos and a method to measure the hotness of video streams in the repository.
- Proposing the management methods of video stream repository to reduce the incurred cost of video stream providers in using cloud services. The methods choose either pre-transcode, re-transcode, or partially pre-transcode video streams in a repository.
- Proposing video streams clustering methods to minimize the overhead time of executing of the management methods video stream repository.
- Analyzing the impact of the management methods of video stream repository on video streams with different access patterns.

Experimental results demonstrate that our proposed the management methods of video stream repository can reduce the incurred cost of using cloud services significantly. Thus, the research outcome of this chapter can help video stream providers to reduce their incurred cost without losing the quality of streaming service they provided.

Our work differs from the previous work [95] in that we develop methods that minimize the incurred cost of cloud services for video streaming by quantifying the hotness of video streams. We also propose videos streams clustering in the repository to

Figure 4.1. different shapes of non-long access for different video streams



minimize the execution time of our methods.

4.2. System Model

For the sake of clarity, the symbols that are used in this chapter are summarized in Table 4.1.

4.2.1. Access Pattern to a Video Stream

Performing pre-transcoding or re-transcoding on a video stream depends on the access pattern to the stream. Therefore, it is crucial to understand the access pattern to video streams. We define *video access rate* as the number of times the video stream is requested by the viewer within a period. However, it does not determine if the requested video stream is watched to the end of it or not. In fact, recent studies (*e.g.,* [13]) reveal that, in a video stream, the beginning GOPs are watched more frequently than the rest of the video stream. Miranda *et al.* [13] show that the distribution of views within a video stream provider generally follows a long-tail distribution. More specifically, they show that the

distribution of accesses to GOPs of a video stream can be expressed by the Power-law [84] model. Based on this model, for a video stream with video access rate υ_i, we can estimate the access rate to the GOPs of the video stream. Let G_{ij} the jth GOP in video stream i. Then, the estimated GOP access rate for G_{ij}, denoted ε_{ij}, is calculated based on Equation 4.1. In this equation, α is called the power coefficient, and its value is 0.1.

$$\varepsilon_{ij} = \upsilon_i \cdot j^{-\alpha} \tag{4.1}$$

Although the GOP access rate for most of the video streams in a repository follows a long-tail pattern, there are video streams whose GOPs' access rate do not follow this pattern [83]. In these video streams, some scenes (GOPs), located in the middle or end of the video stream, are accessed more frequently than other scenes. An example of scenes with tremendously higher access rate can be found in a soccer match where a player scores a goal. We define this type of video streams as non-long-tail access pattern.

Figure 4.1 represents the difference in GOP access rate of a video stream with a long-tail access pattern versus a video with non-long-tail access pattern. In Sub-figure 4.1a, we notice that the GOP access rate consistently decreases for GOPs later in the stream whereas, Sub-figure 4.1b demonstrates that GOP access rate can be significantly different regardless of the position of the GOP in the video stream.

4.2.2. Repository of Video Streams

Video stream providers keep enormous repositories of video streams [133]. For instance, the size of the repository of YouTube is estimated to reach over one billion GB [134]. In fact, storing video repositories in the cloud has become one of the main costs of video stream providers [135]. One main reason to maintain a large repository is to store several

Table 4.1. Symbols used in proposed methods

Symbols	Description
m_i	Total number of GOPs in video i
G_{ij}	GOP number j of video i
υ_i	Number of views for video i
ε_{ij}	Estimated number of views for G_{ij}
ψ_{ij}	Real number of views for G_{ij}
P_S	Price of storing data in cloud (per GB)
P_T	Price of using VM for (per hour)
C_{S_i}	Cost of storing video stream i
C_{T_i}	Cost of transcoding video stream i
τ_i	Transcoding time of video stream i
τ_{ij}	Transcoding time of G_{ij}
$C_{S_{ij}}$	Cost of storing G_{ij}
$C_{T_{ij}}$	Cost of transcoding G_{ij}
R_{ij}	Storage to transcoding cost ratio of G_{ij}
H_i	Hotness of video i
h_{ij}	Hotness of GOP G_{ij}

formats (pre-transcoding) of the same videos.

However, recent studies show that access patterns to videos in a video repository follow a long-tail distribution [13]. This means that few video streams in the repository are watched very frequently (these videos are known as *hot* videos), while the rest of the video streams are rarely watched. It has been reported that in the case of YouTube, only 5% of videos are frequently accessed and the rest are rarely watched [136].

Considering the long-tail access pattern to video streams in a repository, we do not need to pre-transcode all formats for all videos in the repository. Ideally, we should only pre-transcode frequently accessed (*hot*) video streams and the rest of the streams need to be re-transcoded upon requesting them by viewers. However, currently, there is not any method to precisely identify hot videos or to measure the degree of hotness in a repository. In particular, a method is required to measure (quantify) the hotness for a video stream. Then, based on the hotness measure, we can make an appropriate decision either to pre-transcode or re-transcode the video stream.

4.3. Quantifying Hotness of Video Streams

4.3.1. Overview

Our proposed method for quantifying hotness of a video stream relies on the cost of performing pre-transcoding and re-transcoding of the video stream. Thus, in this section, we first introduce methods to calculate these costs in the cloud. Then, we introduce our method to quantify video stream hotness in Sub-section 4.3.4.

4.3.2. Cost of Pre-transcoding

The cost of a pre-transcoding an existing format of a video stream is the cost of storing that format of the video stream. Once the video stream is pre-transcoded and stored, the

number of accesses (views) to the video stream does not change the incurred cost.

In cloud-based video streaming, storage cloud services (*e.g.,* S3 in Amazon Cloud [a]) are used for storing video streams. Let S_i the size of video i in GB and P_S the price of storing one GB of data in the cloud storage. Then, the cost of pre-transcoding for video i, denoted C_{S_i}, is calculated based on Equation 4.2.

$$C_{S_i} = S_i \cdot P_S \tag{4.2}$$

Another cloud service that is commonly used for storing and delivering video streams is CDN (*e.g.,* CloudFront in Amazon Cloud [b]). Video stream providers utilize CDN to minimize the delay in streaming videos. In this case, the cost of pre-transcoding a video stream is dependent on the cost of using storage and CDN, denoted P_{CDN}, and is calculated based on Equation 4.3.

$$C_{S_i} = S_i \cdot (P_S + P_{CDN}) \tag{4.3}$$

4.3.3. Estimated Cost of Re-transcoding

The incurred cost of re-transcoding a video stream depends on the cost of computational services (often, in form of VM) offered by cloud providers. It also depends on the time span of utilizing those VMs, which is generally in an hourly basis. Let P_T the cost of using a VM for an hour, and τ_i the estimated transcoding time of video stream i in the same time unit. Then, the cost of re-transcoding video stream i, denoted C_{T_i}, is obtained using Equation 4.4.

$$C_{T_i} = P_T \cdot \tau_i \tag{4.4}$$

[a] https://aws.amazon.com/s3/

[b] https://aws.amazon.com/cloudfront/

It is noteworthy that Equation 4.4 determines the cost for one time re-transcoding of a video. However, if video i is re-transcoded γ times, the total incurred cost is $\gamma \cdot C_T$.

Estimation of transcoding execution time for video i (τ_i in Equation 4.4) can be achieved based on the past[c] execution information of the same video stream. In particular, the estimated transcoding time of video i with m GOPs is the sum of transcoding times of all the GOPs in that video, *i.e.*, $\tau_i = \sum_{j=1}^{m} \tau_{ij}$, where τ_{ij} is the transcoding time of G_{ij}.

4.3.4. Measuring Hotness of a Video Stream

The way we quantify hotness of a video stream is based on the hotness of its GOPs. Thus, we first define the hotness in the GOP level and then extend that to the video stream level.

Formally, we define a *hot GOP* as a GOP whose estimated cost of re-transcoding is more than the cost of pre-transcoding. To calculate the pre-transcoding and re-transcoding costs at the GOP level, we extend Equations 4.2 and 4.4. Let S_{ij} the size of GOP G_{ij}, then the pre-transcoding cost of G_{ij} is calculated based on Equation 4.5.

$$C_{S_{ij}} = S_{ij} \cdot P_S \tag{4.5}$$

Similarly, the estimated cost of re-transcoding G_{ij}, denoted $C_{T_{ij}}$, is calculated based on Equation 4.6.

$$C_{T_{ij}} = P_T \cdot \tau_{ij} \tag{4.6}$$

Using these concepts, we can introduce the GOP transcoding cost ratio, denoted R_{ij}, based on Equation 4.7.

$$R_{ij} = \frac{C_{S_{ij}}}{\varepsilon_{ij} \cdot C_{T_{ij}}} \tag{4.7}$$

[c] As we are dealing with VOD, we expect that videos have been viewed, thus, transcoded before. However, this is not the case when we deal with live stream videos. In that case, such historic execution information is not available.

Where ε_{ij} is the number of times G_{ij} is viewed. In video streams with long-tail access pattern, the value of ε_{ij} can be estimated based on Equation 4.1. However, for video streams with non-long-tail access pattern, the video stream provider needs to maintain the number of views to each GOP of a video stream.

GOP G_{ij} has to be pre-transcoded, if the cost of re-transcoding is more than pre-transcoding (*i.e.,* $R_{ij} \leq 1$). In this case, G_{ij} is called a *hot* GOP and the value of its hotness, denoted h_{ij}, can be determined based on Equation 4.8.

$$h_{ij} = \begin{cases} 1 & R_{ij} \leq 1 \\ 0 & R_{ij} > 1 \end{cases} \tag{4.8}$$

We define a *hot video stream* as a video stream that needs all of its GOPs be pre-transcoded. However, considering the long-tail access pattern to the GOPs of a video stream (see section 4.2.1), for many video streams, just a portion of that stream can be hot. That is, a video stream can be partially hot and only some GOPs of it need to be pre-transcoded. Hence, we extend the GOP hotness definition to define the hotness value for the whole video stream. Formally, the hotness of video stream i with m GOP denoted H_i is defined based on Equation 4.9.

$$H_i = \frac{\sum_{j=1}^{m} h_{ij}}{m} \tag{4.9}$$

Based on Equation 4.9, the value of hotness H_i is $0 \leq H_i \leq 1$. That means for $H_i = 0$ the video stream i does not include any hot GOP and needs to be completely re-transcoded. Alternatively, $H_i = 1$ indicates that all GOPs are hot. Hence, the whole video is hot and must be pre-transcoded.

4.4. Video Stream Repository Management Methods

4.4.1. Partial Pre-Transconding Methods Based on Long-Tail Access Pattern

Video Level Method

The goal of this algorithm is to minimize the incurred cost of using cloud services for VOD transcoding. For that goal, the algorithm is executed for each video in the VSP repository periodically (*e.g.,* monthly). The pseudo code for this algorithm is presented in Algorithm 1. It receives the video size, estimated video transcoding time, storage unit price, transcoding unit price, and some views of the video in the last period (*e.g.,* last month) as input values. The output of the algorithm is whether to pre-transcode the whole video or re-transcode it upon request.

The algorithm calculates the cost of storage and transcoding according to what was discussed in Equation 4.2 and 4.4 (see steps 1 and 2 Algorithm 1). We calculate the ratio of video storage cost to video transcoding cost in step 6 (called *cost ratio*). When the ratio is less than or equal 1, it means that the cost of pre-transcoding is less than or equal the re-transcoding cost, thus, we store that video format. Otherwise, we call Algorithm 2 to decide about partial pre-transcoding of the video. It is worth noting that for a new video $V = 0$, thus, we postpone its transcoding to when it is requested.

GOP Based Threshold Method

When the cost ratio for a video is greater than 1, Algorithm 2 is called to possibly pre-transcode a portion of the video (termed partially pre-transcoded video). It is proven that the access pattern of GOPs within a video stream follows a long tail distribution [13]. That is, the first GOPs of a video stream are watched more often than the rest. Thus, the cost ratio of GOPs constantly increases for the later GOPs in the video stream.

Algorithm 1: Fully pre-transcode or fully re-transcode a video

Input :

Size of video v: S_v

Transcoding time of video v: τ_v

Transcoding unit price: P_T

Storage unit price: P_S

Number of views in the last time period: V

Output: fully pre-transcoding or fully re-transcode a video

Calculate cost of transcoding of v: $C_T \leftarrow \frac{P_T \cdot \tau_v}{3600}$

Calculate cost of storage of v: $C_S \leftarrow \frac{S_v \cdot P_S}{2^{10}}$

if $V = 0$ **then**

  Re-transcode the whole video v

end

if $\frac{C_S}{V \cdot C_T} \leq 1$ **then**

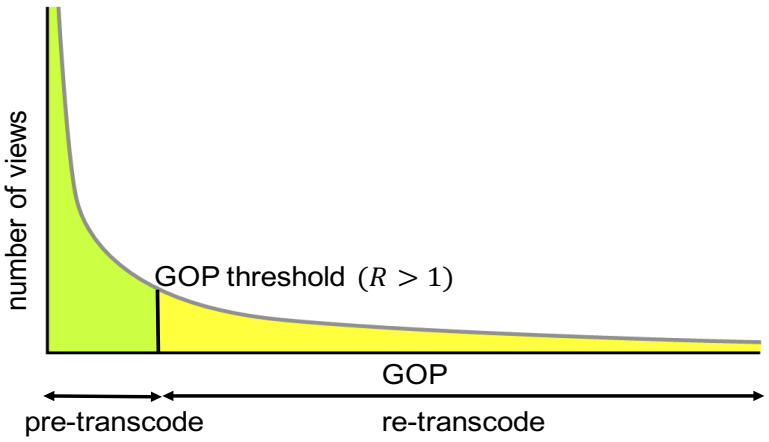
  pre-transcode video v

else

  Perform partial pre-transcoding using Algorithm 2

end

Figure 4.2. Long tail distribution of GOPs views in video stream and partial pre-transcoding a video stream based on GOP threshold



Accordingly, it is possible to find a boundary GOP so that all GOPs before it needs to be pre-transcoded and all GOPs after that re-transcoded. We call this dividing point as GOP threshold (GOP_{th}). Formally, GOP_{th} is defined as the first GOP of a video stream that has its cost ratio greater than one.

Algorithm 2 receives GOPs of a video stream, GOP size, estimated transcoding time of GOP, storage unit price, transcoding unit price and some requests for the video stream in the last period as input values.

Algorithm 2 searches for the GOP threshold by calculating the cost ratio of each GOP sequentially, starting from the first GOP in the video stream (step 1 to 5 in Algorithm 2). After finding the GOP threshold, Algorithm 2 pre-transcodes the GOPs before GOPthreshold and re-transcodes the GOPs after it as shown in Algorimth 2

Fig. 4.2 illustrates how Algorithm 2 functions based on the long tail access pattern to GOPs within a video stream. Once we find the GOP threshold, all the GOPs with cost ratio less than 1 (*i.e.,* $R_i < 1$) are pre-transcoded, and the rest of GOP are re-transcoded.

4.4.2. Partial Pre-Transcoding Methods Based on Generic Access Patterns

Overview

Based on the hotness measure concept discussed in the previous section, we propose methods that for each video in the video stream repository determine which part or parts should be pre-transcoded so that the incurred cost of using cloud services is minimized. These methods are periodically executed on videos in the video stream providers repository (*e.g.,* on a monthly basis) and are called *partial pre-transcoding methods* .

The methods receive view statistics and other meta-data of video streams in the

Algorithm 2: Partial pre-transcoding a video

Input :

video v with m GOP

Size of GOP_i: S_i

Transcoding time of video GOP_i: τ_i

Transcoding unit price: P_T

Storage unit price: P_S

Views number in the last time period : V

Output: Partial pre-transcoding a video

For each GOP_i in video stream v

  Estimated number of views for GOP_i: $P_i \leftarrow V \cdot GOP_i^{-\alpha}$

  Calculate cost of transcoding of GOP_i: $C_{T_i} \leftarrow \frac{P_T \cdot \tau_i}{3600}$

  Calculate cost of storage of GOP_i: $C_{S_i} \leftarrow \frac{S_i \cdot P_S}{2^{10}}$

  Calculate cost ratio of GOP_i: $R_i \leftarrow \frac{C_{S_i}}{P_i \cdot C_{T_i}}$

  if $R_i > 1$ **then**

    $GOP_{th} \leftarrow GOP_i$

    Break;

  else

    Continue;

  end

Pre-transcode GOP_1 to GOP_{th-1}

Re-transcode GOP_{th} to GOP_m

repository, such as GOP size and estimated transcoding time of the GOPs, as input values. The methods are also aware of cloud storage and cloud VM unit prices. In particular, one of the proposed methods (explained in Section 4.4.2.) operates at the video stream level whereas the other one (explained in Section 11) is more granular and operates at the GOP level.

Video-Based Partial Pre-Transcoding Method

The method operates based on the video hotness measure and iterates through all video streams in a repository. For each video, it assumes a long-tail access pattern to GOPs of the video streams in the repository. Hence, the method can estimate the number of views of each GOP by knowing the number access requests of a video stream. The pseudo-code of the method is presented in Algorithm 3.

This method calculates pre-transcoding and re-transcoding costs for each GOP of a video stream based Equations 4.5 and 4.6 (steps 3 and 4). To calculate R_{ij} (in step 6), we need to estimate the number of views of G_{ij} (that is in step 5) based on the number of requests made to video V_i and the long-tail access pattern to GOPs of V_i. Then, the hotness value of each GOP G_{ij} is calculated in step 8. The hotness value for the whole video V_i is calculated based on Equation 4.9 (step 9).

Once hotness value for video stream V_i is calculated, we can determine the portion of the video stream that has to be pre-transcoded. As we assume long-tail access pattern to the video, the portion for pre-transcoding is chosen from the beginning of the video stream V_i. Let H_i the hotness value for video stream V_i with m GOPs, then $\chi_i \leftarrow \lceil H_i \cdot m \rceil$ determines the number of GOPs that needs to be pre-transcoded (step 10 in Algorithm 3).

Algorithm 3: Video-Based partial pre-transcoding method.

Input :

A repository of videos, each video shown as V_i with m $GOPs$

Storage size of G_{ij}: S_{ij}

Transcoding time of G_{ij}: τ_{ij}

re-transcoding unit price: P_T

Storage unit price: P_S

Number of requests to view V_i : υ_i

Output: Partially pre-transcoded version for each video V_i

For each video $V_i \in$ repository

  For each GOP $G_{ij} \in V_i$

    Calculate pre-transcoding cost: $C_{S_{ij}} \leftarrow S_{ij} \cdot P_S$

    Calculate re-transcoding cost: $C_{T_{ij}} \leftarrow \tau_{ij} \cdot P_T$

    Estimated No. of views for G_{ij}: $\varepsilon_{ij} \leftarrow \upsilon_i \cdot G_{ij}^{-\alpha}$

    Calculate cost ratio for G_{ij}: $R_{ij} \leftarrow \dfrac{C_{S_{ij}}}{\varepsilon \cdot C_{T_{ij}}}$

    if $R_{ij} \leq 1$ **then**

      $h_{ij} \leftarrow 1$

  Hotness measure for V_i: $H_i \leftarrow \dfrac{\sum_{j=1}^{m} h_{ij}}{m}$

  No. of GOPs for pre-transcoding: $\chi_i \leftarrow \lceil H_i \cdot m \rceil$

  Pre-transcode GOP_{i1} to $GOP_{i\chi}$ of V_i

GOP-Based Partial Pre-transcoding Method

The idea of this method is to identify hot GOPs within a video stream and pre-transcode only those GOPs. The rest of non-hot (cold) GOPs are re-transcoded upon viewer's request. In this method, we need to know the number of times each GOP has been viewed during the last period. The method does not have any assumption on the access pattern to GOPs in a video stream. Hence, it can cover video streams whose access pattern do not follow long-tail access pattern. Similar to Algorithm 3, the method has to be performed for all video streams in the repository.

The pseudo-code for this method is shown in Algorithm 4. In this method, the transcoding cost-ratio for each GOP G_{ij} is calculated based on Equation 4.7 (steps 3 to 5). If the value of cost-ratio for a GOP is less than or equal to one, it implies that the storage (pre-transcoding) cost for the GOP is less than processing (re-transcoding) it and the GOP need to be pre-transcoded. Otherwise, G_{ij} is re-transcoded.

It is worth noting that, in Algorithm 3, video stream providers does not need to keep track of all view information (*i.e.,* meta-data) for each GOP of the video streams in the repository. In fact, in Algorithm 3, because we assume viewing GOPs of a video stream follows a long-tail distribution, we just need to keep the number of times the video stream is requested. Then, the view information for each GOP can be estimated. However, Algorithm 4 requires views information for each GOP and we need to maintain meta-data information for all GOPs for each video stream in the repository.

Algorithm 4: GOP-Based partial pre-transcoding

Input :

A repository with videos, each video shown as V_i with m $GOPs$

Storage size of G_{ij}: S_{ij}

Transcoding time of G_{ij}: τ_{ij}

re-transcoding unit price: P_T

Storage unit price: P_S

Real number of views for G_{ij} : ψ_{ij}

Output: Partially pre-transcoded version for each video V_i

For each video $V_i \in$ repository

  For each GOP $G_{ij} \in V_i$

    Calculate pre-transcoding cost: $C_{S_{ij}} \leftarrow S_{ij} \cdot P_S$

    Calculate re-transcoding cost:$C_{T_{ij}} \leftarrow P_T \cdot \tau_{ij}$

    Calculate cost ratio: $R_{ii} \leftarrow \dfrac{C_{S_{ij}}}{\psi_{ij} \cdot C_{T_{ij}}}$

    if $R_{ij} \leq 1$ **then**

      $h_{ij} = 1$

      pre-transcode G_{ij}

    else

      $h_{ij} = 0$

      re-transcode G_{ij}

4.5. Experimental Setup

4.5.1. Synthesizing Video Streams

A video stream providers generally has a large video stream repository[d]. However, we do not have access to such repositories for our evaluations. Therefore, in this work, we synthesized a large repository from a set of benchmark videos that we have in our local repository. Our local repository includes 40 videos with various content types and is available publicly[e].

To accurately synthesize video streams, we need to know the distribution and characteristics of video streams in terms of GOP size, GOP transcoding time, and number of GOPs in each video. In order to extract this meta-data, we transcoded the video streams on Amazon EC2[f] and analyzed them.

The result of our analysis, shown in Figure 4.3a and 4.3b, demonstrates that GOP size and number of GOPs follow a Gaussian distribution. The means and standard deviations of the Gaussian distributions are listed in Table 4.2. We also conducted a regression analysis on GOP size and its transcoding time and observed that there is a linear relationship between them (see Figure 4.4). As such, we can derive a linear equation to estimate transcoding time of a GOP based on its size.

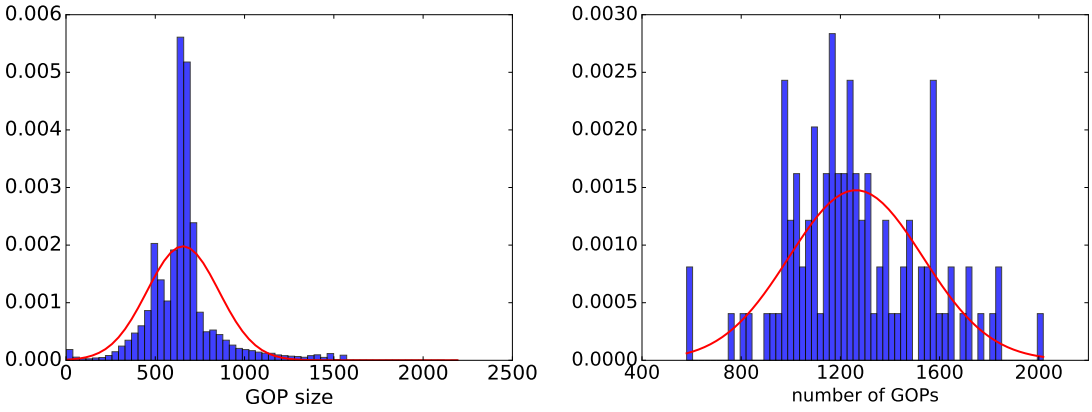
Based on the analyses, we generated the meta-data for 100,000 videos that form our synthesized repository.

[d] YouTube repository is estimated to be one billion GB [134].
[e] The videos can be downloaded from: https://goo.gl/TE5iJ5
[f] https://aws.amazon.com/ec2

Table 4.2. Meta-data of GOP size and number of GOPs extracted from video streams of our repository.

	GOP size (Kbytes)	number of GOPs
Mean	655.08	1262.79
Standard Deviation	201.44	271.46
Total no. of GOPs	124593	130068
Min. GOP size	1.91	580
Max. GOP size	2192.65	2018

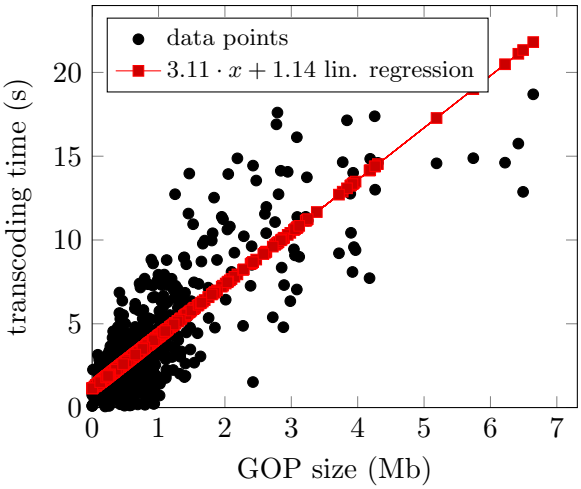
Figure 4.3. Distribution of GOP size and numbers of GOPs in video streams of our repository.



(a) GOP size histogram

(b) GOPs of video histogram

Figure 4.4. Linear regression analysis of GOP size and its transcoding time.



4.5.2. Synthesizing View Information for Video Streams

As mentioned earlier, the access pattern to video streams in a repository follows a long-tail pattern. Thus, similar to Sharma *et al.* [11], we use Weibull distribution to generate the long-tail access pattern to the video streams of our repository. As the percentage of Frequently Accessed Video Streams in a repository is an important parameter in our evaluations, we change it in our synthesized repository by modifying the Shape (α) and Scale (β) parameters in the Weibull distribution. For that purpose, we vary the value of α in the experiment is between $[0.4, 2.4]$, while $\beta = 1$. Figure 4.1a shows an instance of a video stream with long-tail access pattern.

To model video streams that do not follow a long-tail access pattern in the repository, we modify the long-tail access pattern for a portion of the video streams. We determine the number of peak views for a video stream by generating a random number between $[1, 5]$. Each peak view in the video stream has a height and a length that represent the magnitude of view and the width of the peak view, respectively. For instance, Figure 4.1b depicts a video stream with four peak views; each one has a different

Table 4.3. Incurred cost of computation, storage, and CDN in AWS cloud. All costs are in US dollar.

Transcoding	**Storage**	**CDN**
`t2-small`	`S3`	`CloudFront`
$0.026 /hour	$0.03 GB/month	$0.085 GB/month

height and length.

We determine the height of each peak views point by choosing a random number between minimum and maximum access values of that video. The length of each peak view represents the number of GOPs that are accessed within that peak view. To generate the length of each peak view, we used a Normal distribution of $(300, 100)$.

4.5.3. Cost of Cloud Services

All experiments of this research are modeled after Amazon Web Service (AWS) cloud. The costs of AWS cloud storage, CDN, and computational services (*i.e.,* Virtual Machines) used in the experiments are shown in Table 4.3.

4.5.4. Baseline Methods for Comparison

To evaluate our proposed methods, we compare and analyze them against four other methods. These methods are either baseline methods, practically used by video stream providers, or those presented in related research work in the literature. For the sake accuracy of our evaluations, all experiments in this chapter are conducted 50 times, and the mean and 95% of their confidence intervals are reported.

Fully Pre-Transcoding (FPT): This method pre-transcodes all video streams and stores them in the repository before being requested. Currently, this method is widely

used by video stream providers to support different display devices.

Fully Re-Transcoding (FRT): This method re-transcodes video streams each time they are requested by viewers. To the best of our knowledge, video stream providers do not use this method. However, we consider it in our experiment for comparison purposes.

Video-Level Transcoding (VLT): To reduce the incurred cost of using cloud services, Jokhio *et al.* [92] propose a method to either pre-transcode the whole video or re-transcode it. The method works based on the number of views for each video stream. This method is explained in more details in Section 2.4.

GOP-Based Threshold (GBTh): We proposed this method in our earlier work [95]. The method partially pre-transcodes a video stream by finding the first GOP in the video stream (termed threshold GOP) that has its cost ratio greater than 1. GOPs that are located before the threshold GOP are pre-transcoded, and those after the threshold GOP are re-transcoded.

4.6. Performance Evaluations

4.6.1. Impact of Percentage of Frequently Accessed Video Streams in a Repository

In this experiment, the goal is to evaluate the performance of our proposed methods under different access patterns for video streams in the repository. In particular, we are interested to see the impact of different methods on the incurred costs when repositories have different percentages of Frequently Accessed Video Streams .

For that purpose, we generate several video stream repositories with different percentage of Frequently Accessed Video Streams in each repository. Each repository includes 100,000 video streams of out which 30% have non-long-tail access pattern. To vary the percentage of Frequently Accessed Video Streams in each of the synthesized

Table 4.4. Percentage of Frequently Accessed Video Streams (FAVs) in the synthesized repositories by varying the Shape parameter (α) in the Weibull distribution.

	α	**FAVs**
$\beta = 1$	0.4	30%
	0.6	25%
	1	20%
	1.4	15%
	1.8	10%
	2.4	5%

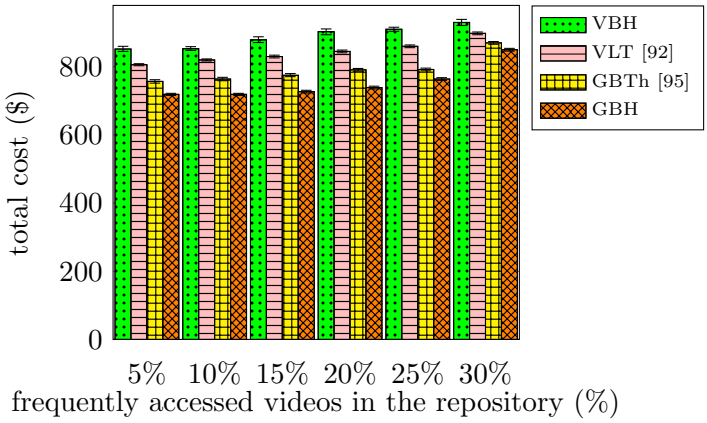
repositories, we change the Shape parameter (α) in the Weibull distribution. The percentage of frequently accessed videos, in each repository, based on the values of α and β are shown in Table 4.4. The average number of views in all synthesized repositories is 1.99.

Figure 4.5a shows the total cost incurred by different methods, explained in Subsection 4.5.4. Vertical axis, in Figure 4.5a, shows the total cost incurred to the stream provider in US Dollar. The total cost, in this figure, includes the cost for pre-transcoding and re-transcoding. The horizontal axis shows repositories with different percentage of Frequently Accessed Video Streams .

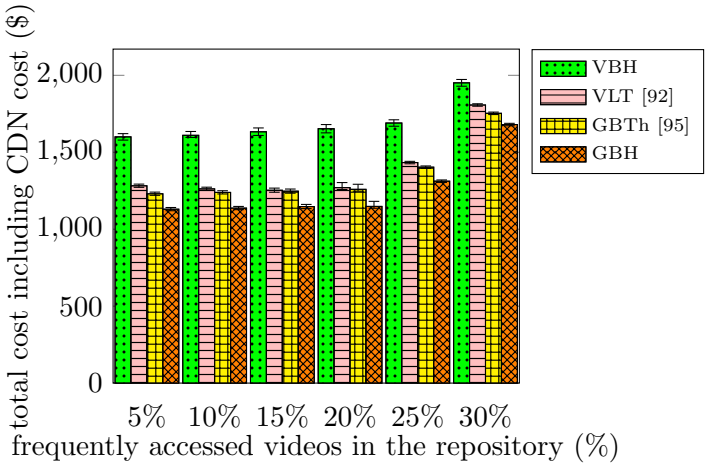
Because the difference of Fully Pre-Transcoding (FPT) and Fully Re-Transcoding FRT methods with other methods is significant, for the sake of better presentation, we do not include them in the figure. In summary, GBH reduces the incurred cost by 3.76X and 10X when compared with FRT and FPT, respectively. Interested readers can refer to the B appendix to see a detailed comparison with these methods.

In Figure 4.5a, we observe that as the percentage of Frequently Accessed Video

Figure 4.5. (a) Incurred costs of different partial pre-transcoding methods (in US Dollar) when the percentage of Frequently Accessed Video Streams varies in the repository. (b) Incurred costs of different partial pre-transcoding methods when cloud storage and cloud CDN is used (in US Dollar). Horizontal axis shows the percentage of Frequently Accessed Video Streams .



(a)



(b)

Streams rises, the overall incurred cost increases. In this figure, we can also observe that GBH method outperforms all other methods. The reason is that GBH checks the hotness for every single GOP of the video streams. GBH outperforms the VLT method [92] by up to 13% when 5% of the video streams in the repository are frequently accessed. However, the difference between GBH and other methods decreases, as the percentage of Frequently Accessed Video Streams in the repository increases. In particular, the outperformance of GBH over VLT reduces to 6% when 30% of video streams in the repository are frequently accessed. This is because when a repository contains a high percentage of Frequently Accessed Video Streams , most of the Frequently Accessed Video Streams are pre-transcoded and partial pre-transcoding methods is applied on few video streams, hence, reducing its impact.

As we can see in Figure 4.5a, GBTh method slightly outperforms VBH. This is because GBTh locates the GOP threshold for pre-transcodes more precisely than the VBH method.

In a similar experiment, we consider the case that the video stream providers uses cloud-based CDN to distribute video streams to the viewers. Figure 4.5b shows the results of the same experiment when the cost of cloud CDN is included in the total incurred cost of the provider. We observe that, in general, the incurred cost has increased due to using CDN. Specifically, methods, such as VBH, that tend to pre-transcode more video streams lead to a remarkably higher incurred cost (between approximately 87% to 100%). In this experiment, again, GBH incurs the minimum cost when compared with other methods. In particular, when 5% of the repository is Frequently Accessed Video Streams , GBH reduces the total incurred a cost by up to 10% and 15%, when compared with GBTh and

VLT, respectively. This shows two to four percent more saving in comparison with other methods when the video stream providers stream their videos through CDN.

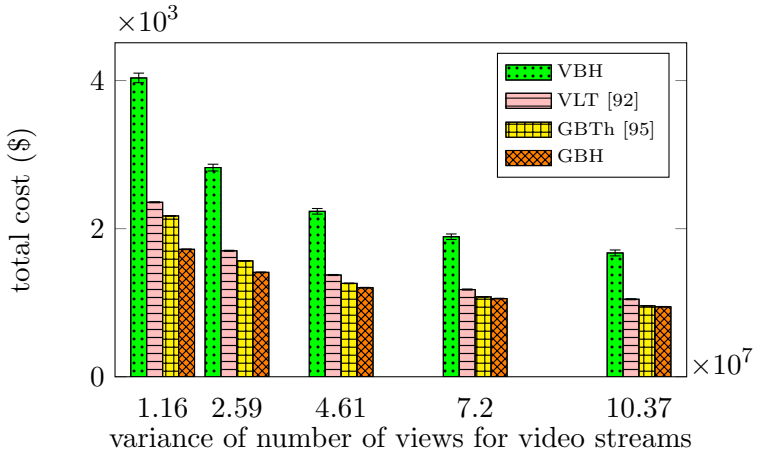
4.6.2. Impact of Increasing Variance of Number of Views in a Repository

As mentioned earlier, the pattern of accessing video streams in a repository follows a long-tail distribution. However, the intensity of the long-tail distribution can vary in different repositories and can impact the incurred cost resulted from various partial pre-transcoding methods. Therefore, in this experiment, we aim to evaluate the behavior of the proposed partial pre-transcoding methods under different long-tail patterns. For that purpose, we alter the variance of a number of views to video streams in the repository. Higher values of variance express a repository in which few videos are viewed very frequently while the rest of video streams are barely viewed. Alternatively, lower values of variance express a repository in which the difference between a number of views of cold and hot videos is low. We alter the values of variance from 1.16×10^7 to 10.37×10^7 and create multiple repositories. Then, for each repository, we estimate the total incurred cost when different partial pre-transcoding methods are used. In all repositories, 20% of videos are considered as hot videos. Also, in all of them, 30% of video streams have non-long-tail access pattern.

Figure 4.6 shows the results of the experiment. In this figure, the horizontal axis shows the variance of a number of views in different repositories and the vertical axis shows the total incurred cost (in US Dollar) when different partial pre-transcoding methods are applied.

In general, we can observe that, in all methods, the total incurred cost decreases as the variance of views increases. The reason is that when the variance is high video stream

Figure 4.6. Incurred costs of different partial pre-transcoding methods when the variance of number of views for video streams increases (in US Dollar).



is either hot or cold. Therefore, they are either fully pre-transcoded or re-transcoded. In other words, when the variance of views in a repository is high, there is not much scope for partial pre-transcoding methods to function.

We also observe that GBH incurs the minimum cost when compared to other methods. However, the difference is more significant when the variance is lower. In particular, when the variance is 1.16×10^7 GBH saves 67% and 21% in the total costs when compared with VBH, and GBTh, respectively. Alternatively, we observe that when the variance is 10.37×10^7, GBH saves 44% when compared with VBH and incurs approximately the same cost when compared with GBTh. The reason for more remarkable cost-saving, when the variance is low, is that a large part of the repository has the potential for partial pre-transcoding. In addition, GBH evaluates each GOP to see the effectiveness of pre-transcoding or re-transcoding. Therefore, it can save more cost than the other methods.

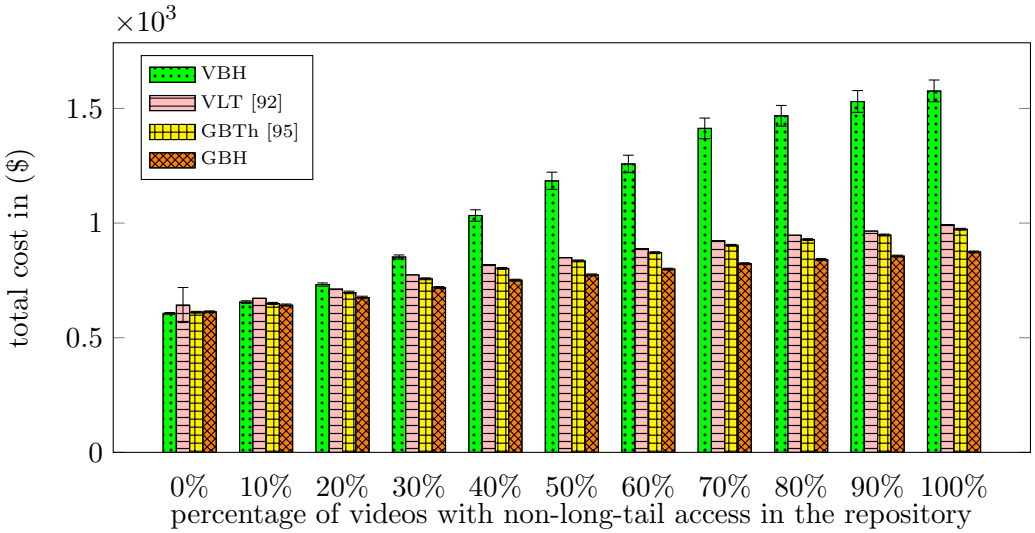
4.6.3. Impact of Percentage of Video streams with Non-Long-Tail Access Pattern in the Repository

As mentioned earlier, the distribution of views within some video streams does not follow the long-tail pattern. That is, there are portions of the video streams that are not necessarily at the beginning of it but receive a large number of views in comparison with the rest of the video stream (see Section 4.2.1). The percentage of video streams with the non-long-tail access pattern, however, can impact the performance of the partial pre-transcoding methods. Hence, we are interested to see how different methods perform when the percentage of video streams with non-long-tail access pattern increases.

In this experiment, we synthesized repositories that include different percentages of video streams with non-long-tail access pattern (see Section 4.5.2) and measured the incurred costs resulted from different partial pre-transcoding methods. Figure 4.7 shows results of the experiment. The horizontal axis, in this figure, shows the percentage of video streams with non-long-tail access pattern and the vertical axis shows the overall incurred cost (in US Dollar).

According to the results, shown in Figure 4.7, the amount of incurred cost, in general, rises as the percentage of video streams with non-long-tail access pattern increases in the repository. However, the rise of the VBH method is more significant than others, from $606 when there is no video stream with non-long-tail pattern to $1,576 when all video streams have non-long-tail pattern. The reason for the remarkable increase is that VBH assumes non-long-tail access pattern for videos and pre-transcodes from the beginning of the streams. For the same reason, when less than 30% of video streams have non-long-tail access pattern, VBH performs similarly to other methods.

Figure 4.7. Incurred costs of different partial pre-transcoding methods (in US Dollar) when the percentage of video streams with non-long-tail access pattern varies in the repository.



In this experiment, we can also observe that GBH has the least increase in the incurred cost (from \$606 to \$874) when compared with other methods. The reason is that GBH does not assume any pattern for the distribution of a number of views within video streams and sweeps all GOPs within the streams to figure out if they should be pre-transcoded or re-transcoded. In particular, the cost-efficiency of GBH is more remarkable in repositories that have a high percentage of video streams with the non-long-tail access pattern. It saves approximately 8% to 10% in the incurred cost, when compared with GBTh, for repositories with more than 70% video stream that has a non-long-tail access pattern. We can conclude that when the percentage of video streams with a non-long-tail access pattern is high and dominates the whole repository, it is worthwhile to investigate the cost-efficiency for each GOP.

4.7. Summary

In this chapter, we proposed partial pre-transcoding methods to efficiently store video streams in repositories in such a way to reduce the cost of using cloud resources. Firstly, we defined what a hot video is and then came up with a method to quantify its hotness. Afterwards, we applied the concept of hotness on video streams to pre-transcode the hot part while re-transcode the remaining part. We evaluated the proposed in different scenarios based on the percentage of frequently accessed video streams in the repositories and on their access.

In the next chapter, we propose a method to cluster the synthesized video streams in repositories, the clustering concept is based on video stream hotness. The clustering method aims to reduce the execution time overhead of the partial pre-transcoding methods.

CHAPTER 5: Low Overhead Partial Pre-transcoding for Video Streaming Repository Management

5.1. Introduction

The partial pre-transcoding methods mentioned in the previous chapter are executed periodically on a video stream repository. However, for a large video stream repository, the methods can take a long time to execute. Thus, the video stream providers have to execute them infrequently. However, as accessing video streams varies over time, the hotness of the video streams changes frequently and partial pre-transcoding methods need to be deployed more often to cope with the variations. Therefore, we need a mechanism to apply partial pre-transcoding methods more efficiently by reducing their execution times. In this section, we propose a mechanism that reduces the execution time of the partial pre-transcoding methods without any major impact on their performance.

Partial pre-transcoding methods impact only video streams that are partially hot. Therefore, if we can exclude partially hot video streams and apply the partial pre-transcoding methods only on them, then the overall execution time is reduced.

5.2. Clustering Video Streaming Repository Based on Hotness

Our proposed mechanism works based on clustering of the video streams according to their number of views. The goal of clustering is to separate hot videos and cold videos from the rest of video streams in the repository.

We use the number of views of a video stream as a clustering parameter. We utilize Jenks Natural Breaks Optimization algorithm [137], which is a one-dimensional clustering (grouping) method, to group video streams that require a similar kind of re-transcoding or pre-transcoding operation. Jenks Natural Breaks is an iterative algorithm that aims to

maximize the Goodness Of Variance (GOV) fit and is defined based on Equation 5.1.

$$GOV = \frac{SDV - SDC}{SDV} \tag{5.1}$$

Where SDV is defined as the sum of squared deviations of views of video streams in a repository and is calculated based on Equation 5.2. To maximize GOV, we only need to minimize the sum of squared deviations for each cluster.

$$SDV = \sum_{i=1}^{N} (\upsilon_i - \overline{\upsilon_i})^2 \tag{5.2}$$

where υ_i is the number of views of video stream i; also, N is the total number of video streams in a repository; and $\overline{\upsilon_i} = \frac{\sum_{i=1}^{N} \upsilon_i}{N}$. SDC is defined as the sum of squared deviations of video stream views across clusters and is calculated based on Equation 5.3.

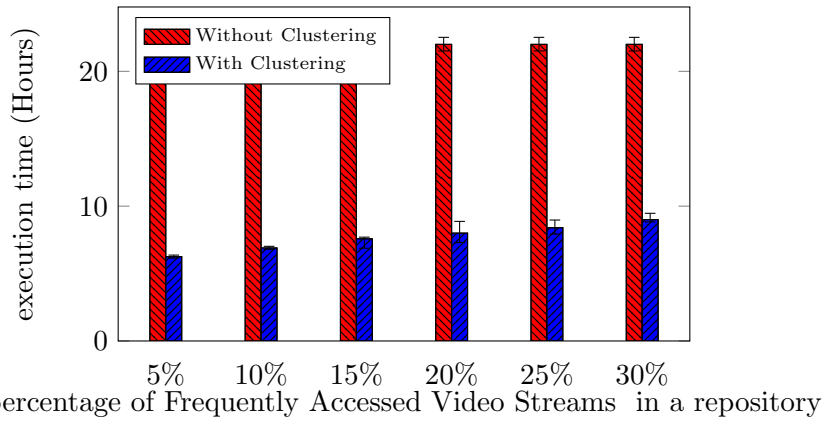
$$SDC = \sum_{\kappa=1}^{c} (V_\kappa - \overline{V_\kappa})^2 \tag{5.3}$$

where c is the number of clusters that are chosen initially.

Accordingly, the algorithm starts with a random clustering and iteratively calculates the mean and squared deviation for each cluster. At the end of each iteration, the algorithm relocates the elements in the cluster with the maximum squared deviation to the cluster with the minimum squared deviation. Then, for convergence (*i.e.,* termination), the algorithm checks if the total squared deviation (SDV) for the whole set is minimized.

Depending on the distribution of views in a given video stream repository, Jenks Natural Breaks algorithm splits the repository into a different numbers of clusters. However, we need to reorganize the generated clusters into three groups (*i.e.,* cold, hot, and partially hot video streams). For that purpose, we iteratively merge neighboring

Figure 5.1. Execution time overhead of partial pre-transcoding methods (in hours) with and without clustering video streams in the repository.



clusters (*i.e.,* clusters with succeeding means) generated by Jenks Natural Breaks and configure them into three clusters. Then, for each configuration, we estimate the total incurred cost of pre-transcoding and re-transcoding. The configuration with the minimum cost is chosen as the final clustering for the video streams in the repository.

5.3. Evaluating the Impact of Clustering on Partial Pre-Transcoding Method

Although GBH, in general, outperforms other partial pre-transcoding methods that were studied in the previous subsections, it imposes the highest execution time overhead. The reason for its high overhead is processing each GOP for each video stream in the repository. Therefore, our goal, in this experiment, is to measure the effectiveness of the clustering method (presented in Section 5.2) on reducing the execution time overhead of the GBH method.

To measure the effectiveness of the clustering method, we conduct an experiment by generating repositories with varying percentage of Frequently Accessed Video Streams in them. For each repository, we measure the execution time overhead of the GBH method in two scenarios: (A) when the clustering method is applied and (B) without the

clustering method.

Figure 5.1 shows the results of the experiment. The horizontal axis, in this figure, represents the percentage of Frequently Accessed Video Streams in the repository and the vertical axis represents the execution time overhead of the GBH method. As we expected, when the clustering method is not applied, the execution time overhead of GBH does not change for different percentages of Frequently Accessed Video Streams. This is because the method is carried out on the GOPs of each video stream, regardless of its hotness. We observe that the execution time of GBH decreases substantially when the clustering method is applied. In particular, for 5% and 30% of Frequently Accessed Video Streams , the execution time overhead is reduced by 72% and 59%, respectively. As we move to repositories with a higher percentage of Frequently Accessed Video Streams, we observe a slight increase in the execution time overhead, when the clustering method is applied. The reason for the increase is that in repositories with a higher percentage of Frequently Accessed Video Streams, the cluster of video streams that GBH should be applied on becomes bigger, hence, its execution time overhead increases.

5.4. Summary

In this chapter, we proposed a method to cluster videos streams in the repository so that the partial pre-transcoding methods are applied only on video streams that are partially hot. When applying the proposed clustering method on video streams in the repository and then applying GOP-based hotness method on the clustered repository, the experimental results show that the execution time overhead of GOP-based hotness method is reduced by up to 72%.

In the next chapter, we conclude the works of this dissertation and then discuss

the research future of video streaming in the cloud.

CHAPTER 6: Conclusion and Future Works

6.1. Conclusion

In this dissertation, we spotlighted the technology of video streaming, including its challenges and advantages, were addressed. Moreover, cloud-based video streaming provides innovative solutions to stream videos, particularly, cloud resources become the feasible solution for video streams providers to process their videos in a less costly and more time-efficient manner.

Using cloud services, we analyzed video transcoding in terms of quality of service and cost. More specifically, we came up with a model called performance cost trade-off. Based on criteria and requirements of users, the model assigns optimally the transcoding tasks of video streams to the heterogeneous cloud VMs. Furthermore, we observed a correlation between transcoding time and GOP size, which is the essential clue to synthesize video streams repositories.

Since the video streams repositories contain an enormous number of videos, they incur a high cost of maintenance. Thus, we proposed management methods to reduce the incurred cost of cloud-based video streaming through pre-transcoding, re-transcoding, or partially pre-transcoding of video streams. For that purpose, we needed to measure the hotness of the video streams. Accordingly, in this dissertation, we proposed a method to quantify the hotness of video streams. The hotness measure was then used to develop repository management methods that are executed periodically and determined either to pre-transcode, re-transcode, or partially pre-transcode video streams.

We analyzed the impact of our proposed methods by synthesizing different repositories with various characteristics. Experiment results demonstrated that as the

percentage of Frequently Accessed Video Streams increases in a repository, the GBH method reduces the total incurred cost (approximately 12%) when compared to previous works in the area. Furthermore, when the variance in a number of views of video streams increases, the GBH method reduces the total incurred cost by up to 10%. We conclude that GBH is specifically effective when the repository includes a high percentage of videos with a non-long-tail access pattern. Although partial pre-transcoding methods are executed periodically, their execution time overhead for large repositories is high. Therefore, in this research, we also presented a method to cluster video streams based on their hotness values. Then, a repository management method is only applied on a small subset of video streams in the repository. We observed that the clustering method reduces the execution time overhead of the repository management methods by up to 71.5%.

6.2. Future of Cloud-based Video Streaming Research

Although cloud services have been useful in obviating many technical challenges of video streaming, there are still areas that require further exploration by researchers and practitioners. In particular, we believe that addressing the following areas will be impactful in the cloud-based video streaming industry:

- **Interactive video streaming using clouds.** Interactive video streaming is defined as to provide the ability for video stream providers to offer any form of processing, enabled by video stream providers, on the videos being streamed [138, 139]. For instance, a video stream provider may need to offer video stream summarization service for its viewers. Another example can be providing a service that enables viewers to choose subtitles of their languages. Although some of

these interactions are currently provided by video stream providers[a], there is still not a generic video streaming engine that can be extended with new streaming services by the video provider.

Since there is a wide variety of possible interactions with video streams, it is not possible to pre-process and store video streams; instead, they have to be processed upon viewers' request and in a real-time manner. It is also not possible to process the video streams on viewers' thin-clients (e.g., smart-phones), due to energy and compute limitations [140]. Providing such a streaming engine entails answering several research and implementation problems including:

- How to come up with a system for the video streaming engine that is pluggable and a high-level language for users to extend the engine with their desired services and without any major programming effort?
- How does the streaming engine accommodate a new service? That implies asking how the streaming engine can learn the execution time distribution of tasks for user-defined interactions? How can the streaming engine provision cloud VMs and schedule video streaming tasks on the allocated VMs so that the Quality of Service (QoS) for viewers is guaranteed and the minimum cost is incurred by the video stream provider?

- **Harnessing heterogeneity in cloud services for video stream processing.** Current cloud-based video stream providers utilize homogeneous cloud VMs for video stream processing [10, 9]. However, cloud providers offer heterogeneity for each cloud service [141]. For instance, they provide heterogeneous VM types, such as

[a]For instance, YouTube allows viewers to choose their preferred spatial resolution

GPU, CPU-Optimized, Memory-Optimized, and IO-Optimized in Amazon cloud [141]. The same is with heterogeneous storage services (*e.g.,* SSD and HDD storages).

The question that arises how we can harness cloud services to offer video streaming with QoS considerations and with the minimum incurred cost for the video stream provider? How can we learn the affinity of a user-defined interaction with the heterogeneous services? How can we strike a trade-off between the performance offered and incurred cost of heterogeneous cloud services?

- **Supporting live-streaming and VOD.** Live-streams and VODs are structurally similar and also have similar computational demands. However, processing them is not entirely the same. In particular, live-streaming tasks have a hard deadline and have to be dropped if they miss their deadline [9] whereas VOD tasks have a soft deadline and can tolerate deadline misses. In addition, there is no historic computational information for live-streaming tasks to estimate the execution time of new tasks. This is not the case for VOD tasks.

  Based on these differences, video stream providers utilize different video streaming engines to provide both services [88, 9]. The question is how we can have an integrated streaming engine that can accommodate both of the streaming types simultaneously? How will this affect the scheduling and processing time estimation of the streaming engine?

- **Prediction of Number of Views for Video Stream.** The number of views of a video stream is one of the major factors that play a role in determining the hotness of a video stream. However, predicting or estimating the number of views can help

video streams providers to anticipate the video streams that should be stored and those that should be transcoded upon request. The prediction of a number of views can be done using machine learning techniques on previous views of the video streams.

- **Cloud-Based Video Caching for Cost-Efficient Repository Management.** Cloud resources offer a diversity of storages with different prices for video streaming. For instance, AWS cloud provides S3, EC2, and EBS storages, in addition to CloudFront (CDN). These storages have different latencies. Video streams can be pre-transcoded in hierarchical storage and a cost-efficient manner. That is, the high accessed video streams could be pre-transcoded in the storage which has low latency, and the video streams with different rates of access should be pre-transcoded into storages with different latencies. The questions that arise: under which cloud, and which type of service (*i.e.,* storage and CDN) should a video stream be pre-transcoded? Thus, the goal is to stream videos to users with minimum delay and with effective cost.

- **Cloud-Based Video Summarization.** The number of users for video streaming has increased tremendously. Consequently, the processing servers are limited when all users concurrently request the same video stream for watching. As a result, the users' requests make the system oversubscribed. In such case, the requested video is streamed with time delay, and the user may stop watching the video stream. The question is how we can overcome oversubscription without expending more money? The goal of this research is to determine how to maintain user satisfaction in a cost-efficient manner.

One of the solutions can be by providing a ready version (*i.e.,* summarized version) of the video stream for the users in case of network oversubscription. This version should represent an overall idea about the original version of the video stream. When the network is oversubscribed, the system stops processing the requests of users whose memberships are free and thus provides them with the summarized version, and thus the oversubscription is mitigated. While the users who have a premium membership can watch the video stream in real time.

BIBLIOGRAPHY

[1] S.-H. Liu, H.-L. Liao, and J. A. Pratt, "Impact of media richness and flow on e-learning technology acceptance," *Computers and Education*, vol. 52, no. 3, pp. 599–607, 2009.

[2] C.-W. Lin, Y.-C. Chen, and M.-T. Sun, "Dynamic region of interest transcoding for multipoint video conferencing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 10, pp. 982–992, 2003.

[3] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt *et al.*, "A system for video surveillance and monitoring," *VSAM final report*, pp. 1–68, 2000.

[4] H. Joo, H. Song, D.-B. Lee, and I. Lee, "An effective iptv channel control algorithm considering channel zapping time and network utilization," *IEEE Transactions on broadcasting*, vol. 54, no. 2, pp. 208–216, 2008.

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

[6] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, "Video transcoding: an overview of various techniques and research issues," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 793–804, Oct. 2005.

[7] "Netflix," http://techblog.netflix.com/2012/12/videos-of-netflix-talks-at-aws-reinvent.html, accessed: 2017.

[8] X. Li, M. A. Salehi, and M. Bayoumi, "Cloud-based video streaming for energy-and compute-limited thin clients," in *The Stream Workshop at Indiana University*, 2015.

[9] ——, "VLSI: Video Live Streaming Using Cloud Services," in *Proceedings of the 6th IEEE International Conference on Big Data and Cloud Computing Conference*, ser. BDCloud '16, Oct. 2016.

[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*, ser. CCGrid '16, May 2016.

[11] 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.

[12] "Amazon," https://aws.amazon.com/ec2/pricing/on-demand/, accessed: 2017.

[13] L. C. Miranda, R. L. Santos, and A. H. Laender, "Characterizing video access patterns in mainstream media portals," in *Proceedings of the 22nd International Conference on World Wide Web*, pp. 1085–1092, 2013.

[14] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang, "Unreeling netflix: Understanding and improving multi-cdn movie delivery," in *Proceedings of the IEEE Conference on Computer and Communications Societies*, pp. 1620–1628, 2012.

[15] N. Ramzan, H. Park, and E. Izquierdo, "Video streaming over p2p networks: Challenges and opportunities," *Signal Processing: Image Communication*, vol. 27, no. 5, pp. 401–411, 2012.

[16] 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, Aug. 2013.

[17] T. Deneke, H. Haile, S. Lafond, and J. Lilius, "Video transcoding time prediction for proactive load balancing," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, July 2014.

[18] F. Jokhio, T. Deneke, S. Lafond, and J. Lilius, "Analysis of video segmentation for spatial resolution reduction video transcoding," in *Proceedings of the IEEE International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS)*, pp. 1–6, Dec. 2011.

[19] 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, May 2013.

[20] 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 IEEE Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 254–261, Feb. 2013.

[21] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *IEEE Transactions on circuits and systems for video technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[22] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje, "The latest open-source video codec vp9-an overview and preliminary results," in *Proceeding of the IEEE Conference on Picture Coding Symposium (PCS)*, pp. 390–393, 2013.

[23] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1649–1668, 2012.

[24] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol–http/1.1," Tech. Rep., 1999.

[25] B. Lesser, *Programming flash communication server.* " O'Reilly Media, Inc.", 2005.

[26] H. Schulzrinne, "Real time streaming protocol (rtsp)," 1998.

[27] R. Buyya, M. Pathan, and A. Vakali, *Content delivery networks.* Springer Science Business Media, vol. 9, 2008.

[28] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," *Peer-to-peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, 2008.

[29] E. Latja, "Parallel acceleration of h.265 video processing," Ph.D. dissertation, Aalto University, 2017.

[30] D. Wu, Z. Xue, and J. He, "icloudaccess: Cost-effective streaming of video games from the cloud with low latency," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 8, pp. 1405–1416, Aug. 2014.

[31] F. Lao, X. Zhang, and Z. Guo, "Parallelizing video transcoding using map-reduce-based cloud computing," in *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 2905–2908, May 2012.

[32] 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, Mar. 2003.

[33] O. Werner, "Requantization for transcoding of mpeg-2 intraframes," *IEEE Transactions on Image Processing*, vol. 8, pp. 179–191, Feb. 1999.

[34] N. Bjork and C. Christopoulos, "Transcoder architectures for video coding," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 1, pp. 88–98, Feb. 1998.

[35] S. Goel, Y. Ismail, and M. Bayoumi, "High-speed motion estimation architecture for real-time video transmission," *The Computer Journal*, vol. 55, no. 1, pp. 35–46, Apr. 2012.

[36] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital video: an introduction to MPEG-2*. Springer Science Business Media, 1996.

[37] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[38] M. Shaaban and M. Bayoumi, "A low complexity inter mode decision for mpeg-2 to h. 264/avc video transcoding in mobile environments," in *Proceedings of the 11th IEEE International Symposium on Multimedia (ISM)*, pp. 385–391, Dec. 2009.

[39] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J. C. Chen, "A survey of energy efficient network protocols for wireless networks," *wireless networks*, vol. 7, no. 4, pp. 343–358, 2001.

[40] S. Balcisoy, M. Karczewicz, and T. Capin, "Progressive downloading of timed multimedia content," uS Patent App. 10/865,670, Jun. 2004.

[41] L. Parziale, W. Liu, C. Matthews, N. Rosselot, C. Davis, J. Forrester, D. T. Britt *et al.*, *TCP/IP tutorial and technical overview*. IBM Redbooks, 2006.

[42] H. Zheng and J. Boyce, "An improved udp protocol for video transmission over internet-to-wireless networks," *IEEE Transactions on Multimedia*, vol. 3, no. 3, pp. 356–365, 2001.

[43] V. N. Padmanabhan and J. C. Mogul, "Improving http latency," *Computer Networks and ISDN Systems*, vol. 28, no. 1-2, pp. 25–35, 1995.

[44] V. Jacobson, R. Frederick, S. Casner, and H. Schulzrinne, "Rtp: A transport protocol for real-time applications," 2003.

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

[46] A. Begen, T. Akgul, and M. Baugher, "Watching video over the web: Part 1: Streaming protocols," *Journal of IEEE Internet Computing*, vol. 15, no. 2, pp. 54–63, 2011.

[47] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An evaluation of bitrate adaptation methods for http live streaming," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 693–705, 2014.

[48] N. Bouzakaria, C. Concolato, and J. Le Feuvre, "Overhead and performance of low latency live streaming using mpeg-dash," in *Proceedings of The 5th International Conference on Information, Intelligence, Systems and Applications, IISA 2014,*, pp. 92–97, 2014.

[49] C. Timmerer and C. Müller, "Http streaming of mpeg media," *Streaming Day*, 2010.

[50] C. Hanna, D. Gillies, E. Cochon, A. Dorner, J. Alred, and M. Hinkle, "Demultiplexer ic for mpeg2 transport streams," *IEEE Transactions on Consumer Electronics*, vol. 41, no. 3, pp. 699–706, 1995.

[51] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An analysis of internet content delivery systems," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 315–327, Dec. 2002.

[52] A. Vakali and G. Pallis, "Content delivery networks: status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, Nov 2003.

[53] "akamai," http://www.akamai.com/, accessed: 2017.

[54] "level3," http://www.leveI3.com/, accessed: 2017.

[55] J. Apostolopoulos, T. Wong, W.-t. Tan, and S. Wee, "On multiple description streaming with content delivery networks," in *Proceedings of the 21th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3, pp. 1736–1745, 2002.

[56] C. D. Cranor, M. Green, C. Kalmanek, D. Shur, S. Sibal, J. E. Van der Merwe, and C. J. Sreenan, "Enhanced streaming services in a content distribution network," *IEEE Internet Computing*, vol. 5, no. 4, pp. 66–75, 2001.

[57] S. Wee, J. Apostolopoulos, W.-t. Tan, and S. Roy, "Research and design of a mobile streaming media content delivery network," in *Proceedings of the IEEE International Conference on Multimedia and Expo, ICME'03.*, vol. 1, pp. I–5, 2003.

[58] L. Kontothanassis, R. Sitaraman, J. Wein, D. Hong, R. Kleinberg, B. Mancuso, D. Shaw, and D. Stodolsky, "A transport layer for live streaming in a content delivery network," *Proceedings of the IEEE 92*, no. 9, pp. 1408–1419, 2004.

[59] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, "Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky," in *Proceedings of the 17th ACM international conference on Multimedia*, pp. 25–34, 2009.

[60] Z. Zhuang and C. Guo, "Building cloud-ready video transcoding system for content delivery networks (cdns)," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM).*, pp. 2048–2053, 2012.

[61] D. K. Krishnappa, M. Zink, and R. K. Sitaraman, "Optimizing the video transcoding workflow in content delivery networks," in *Proceedings of the 6th ACM Multimedia Systems Conference*. ACM, pp. 37–48, 2015.

[62] X. Zhang, J. Liu, B. Li, and Y.-S. Yum, "Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3, pp. 2102–2111, 2005.

[63] Y.-h. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast (keynote address)," in *Proceedings of the ACM SIGMETRICS Performance Evaluation Review*, vol. 28, no. 1, pp. 1–12, 2000.

[64] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek *et al.*, "Overcast: reliable multicasting with on overlay network," in *Proceedings of the 4th Conference on Symposium on Operating System Design & Implementation-Volume 4*. USENIX Association, pp. 14, 2000.

[65] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: A comparative study of live p2p streaming approaches," in *Proceeding of the 26th IEEE International Conference on Computer Communications.*, pp. 1424–1432, 2007.

[66] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast," in *Proceedings of the 2006 14th IEEE International Conference on Network Protocols, ICNP'06.*, pp. 2–11, 2006.

[67] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "Bitos: Enhancing bittorrent for supporting streaming applications," in *Proceedings of the 25th IEEE International Conference on Computer Communications.*, pp. 1–6, 2006.

[68] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2cast: peer-to-peer patching scheme for vod service," in *Proceedings of the 12th ACM international conference on World Wide Web*, pp. 301–309, 2003.

[69] C. Xu, G.-M. Muntean, E. Fallon, and A. Hanley, "A balanced tree-based strategy for unstructured media distribution in p2p networks," in *Proceedings of the IEEE International Conference on Communications, 2008. ICC'08.*, pp. 1797–1801, 2008.

[70] W.-P. K. Yiu, X. Jin, and S.-H. G. Chan, "Vmesh: Distributed segment storage for peer-to-peer interactive video streaming," *IEEE journal on selected areas in communications*, vol. 25, no. 9, 2007.

[71] B. Cheng, H. Jin, and X. Liao, "Supporting vcr functions in p2p vod services using ring-assisted overlays," in *Proceeding of The EEE International Conference on Communications, ICC'07*, pp. 1698–1703, 2007.

[72] T. Xu, J. Chen, W. Li, S. Lu, Y. Guo, and M. Hamdi, "Supporting vcr-like operations in derivative tree-based p2p streaming systems," in *Proceedings of The IEEE International Conference on Communications, ICC'09.*, pp. 1–5, 2009.

[73] M. Garmehi, M. Analoui, M. Pathan, and R. Buyya, "An economic replica placement mechanism for streaming content distribution in hybrid cdn-p2p networks," *Computer Communications*, vol. 52, pp. 60–70, 2014.

[74] D. Xu, S. S. Kulkarni, C. Rosenberg, and H.-K. Chai, "Analysis of a cdn–p2p hybrid architecture for cost-effective streaming media distribution," *Multimedia Systems*, vol. 11, no. 4, pp. 383–399, 2006.

[75] P. J. Shenoy and H. M. Vin, "Efficient striping techniques for variable bit rate continuous media file servers," *Performance Evaluation*, vol. 38, no. 3, pp. 175–199, 1999.

[76] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming video over the internet: approaches and directions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282–300, Mar 2001.

[77] Z. Miao and A. Ortega, "Scalable proxy caching of video under storage constraints," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1315–1327, Sep 2002.

[78] A. M. Berger, "Privacy mode for acquisition cameras and camcorders," uS Patent 6,067,399, May 2000.

[79] A. Senior, S. Pankanti, A. Hampapur, L. Brown, Y.-L. Tian, A. Ekin, J. Connell, C. F. Shu, and M. Lu, "Enabling video privacy through computer vision," *IEEE Security & Privacy*, vol. 3, no. 3, pp. 50–57, 2005.

[80] A. Erdélyi, T. Barát, P. Valet, T. Winkler, and B. Rinner, "Adaptive cartooning for privacy protection in camera networks," in *Proceedings of the 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 44–49, 2014.

[81] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs," *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 2001–2014, 2013.

[82] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *Proceedings of the ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 362–373, 2011.

[83] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 1–14, 2007.

[84] M. E. Newman, "Power laws, pareto distributions and zipf's law," *Contemporary physics*, vol. 46, no. 5, pp. 323–351, 2005.

[85] M. Darwich, M. A. Salehi, E. Beyazit, and M. Bayoumi, "Cost efficient cloud-based video streaming based on quantifying video stream hotness," *Submitted to The Computer Journal*, 2017.

[86] M. Kim, Y. Cui, S. Han, and H. Lee, "Towards efficient design and implementation of a hadoop-based distributed video transcoding system in cloud computing environment," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 8, no. 2, pp. 213–224, Mar. 2013.

[87] 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

Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 482–489, May 2013.

[88] X. Li, M. A. Salehi, and M. Bayoumi, "High Perform On-Demand Video Transcoding Using Cloud Services," in *Proceedings of the 16th ACM/IEEE International Conference on Cluster Cloud and Grid Computing*, ser. CCGrid '16, May 2016.

[89] 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 *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*, pp. 33–38, June 2012.

[90] C.-F. Lai, H.-C. Chao, Y.-X. Lai, and J. Wan, "Cloud-assisted real-time transrating for http live streaming," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 62–70, July 2013.

[91] T. C. Thang, Q.-D. Ho, J. W. Kang, and A. T. Pham, "Adaptive streaming of audiovisual content using mpeg dash," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 1, pp. 78–85, Mar. 2012.

[92] F. Jokhio, A. Ashraf, S. Lafond, and J. Lilius, "A computation and storage trade-off strategy for cost-efficient video transcoding in the cloud," in *Proceedings of the 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 365–372, Sept. 2013.

[93] H. Zhao, Q. Zheng, W. Zhang, B. Du, and Y. Chen, "A version-aware computation and storage trade-off strategy for multi-version VoD systems in the cloud," in

Proceedings of the IEEE Symposium on Computers and Communication (ISCC), pp. 943–948, July 2015.

[94] H. Zhao, Q. Zheng, W. Zhang, B. Du, and H. Li, "A segment-based storage and transcoding trade-off strategy for multi-version vod systems in the cloud," *IEEE Transactions on Multimedia*, vol. 19, no. 1, pp. 149–159, Jan 2017.

[95] M. Darwich, E. Beyazit, M. A. Salehi, and M. Bayoumi, "Cost efficient repository management for cloud-based on-demand video streaming," in *Proceedings of the 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, April 2017.

[96] C. Timmerer, D. Weinberger, M. Smole, R. Grandl, C. Müller, and S. Lederer, "Live transcoding and streaming-as-a-service with mpeg-dash," in *Proceedings of the IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pp. 1–4, June 2015.

[97] D. Seo, J. Kim, and I. Jung, "Load distribution algorithm based on transcoding time estimation for distributed transcoding servers," in *Proceedings of the International Conference on Information Science and Applications*, pp. 1–8, Apr. 2010.

[98] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Transactions on Parallel and Distributed systems*, vol. 22, no. 6, pp. 931–945, June 2011.

[99] G. Lee and R. H. Katz, "Heterogeneity-aware resource allocation and scheduling in the cloud," in *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'11, 2011, pp. 4–4, Oct. 2011.

[100] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," in *Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 159–168, Nov. 2010.

[101] J. Dejun, G. Pierre, and C.-H. Chi, "Ec2 performance analysis for resource provisioning of service-oriented applications," in *Proceedings of the 7th International Joint Conference on Service Oriented Computing*. Springer, pp. 197–207, Nov. 2009.

[102] B. Farley, A. Juels, V. Varadarajan, T. Ristenpart, K. D. Bowers, and M. M. Swift, "More for your money: exploiting performance heterogeneity in public clouds," in *Proceedings of The 3rd ACM Symposium on Cloud Computing*, pp. 20, Nov. 2012.

[103] R. L. Myers, P. Ranganathan, I. Chvets, and K. Pakulski, "Methods and systems for real-time transmuxing of streaming media content," uS Patent 9,712,887, July 2017.

[104] J. Gorostegui, A. Martin, M. Zorrilla, I. Alvaro, and J. Montalban, "Broadcast delivery system for broadband media content," in *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–9, 2017.

[105] F. Chen, K. Guo, J. Lin, and T. La Porta, "Intra-cloud lightning: Building cdns in the cloud," in *Proceedings of the IEEE Conference on Computer and Communications Societies*, pp. 433–441, 2012.

[106] H. Hu, Y. Wen, T.-S. Chua, J. Huang, W. Zhu, and X. Li, "Joint content replication and request routing for social video distribution over cloud cdn: A community clustering method," *IEEE transactions on circuits and systems for video technology*, vol. 26, no. 7, pp. 1320–1333, 2016.

[107] H. Hu, Y. Wen, T.-S. Chua, Z. Wang, J. Huang, W. Zhu, and D. Wu, "Community based effective social video contents placement in cloud centric cdn network," in *proceeding of The IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2014.

[108] Y. Jin, Y. Wen, G. Shi, G. Wang, and A. V. Vasilakos, "Codaas: An experimental cloud-centric content delivery platform for user-generated contents," in *Proceeding of The IEEE International Conference on Computing, Networking and Communications (ICNC)*, pp. 934–938, 2012.

[109] H. Li, L. Zhong, J. Liu, B. Li, and K. Xu, "Cost-effective partial migration of vod services to content clouds," in *Proceeding of the IEEE International Conference on Cloud Computing (CLOUD)*, pp. 203–210, 2011.

[110] Y. Li, Y. Shen, and Y. Liu, "Utilizing content delivery network in cloud computing," in *Proceeding of the IEEE International Conference on Computational Problem-Solving (ICCP)*, pp. 137–143, 2012.

[111] D. A. Rodrguez-Silva, L. Adkinson-Orellana, F. J. Gonz'lez-Castao, I. Armio-Franco, and D. Gonz'lez-Martnez, "Video surveillance based on cloud storage," pp. 991–992, June 2012.

[112] 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," *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1286–1296, Aug 2015.

[113] H. Zhao, Q. Zheng, W. Zhang, B. Du, and Y. Chen, "A version-aware computation and storage trade-off strategy for multi-version vod systems in the cloud," in *Proceedings of the IEEE Symposium on Computers and Communication (ISCC)*, pp. 943–948, July 2015.

[114] H. Zhang, G. Jiang, K. Yoshihira, and H. Chen, "Proactive workload management in hybrid cloud computing," *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 90–100, March 2014.

[115] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena, "Intelligent workload factoring for a hybrid cloud computing model," in *proceedings of the World Conference on Services - I*, pp. 701–708, July 2009.

[116] I. Trajkovska, J. Salvachua Rodriguez, and A. Mozo Velasco, "A novel p2p and cloud computing hybrid architecture for multimedia streaming with qos cost functions," in *Proceedings of the 18th ACM International Conference on Multimedia*, ser. MM '10, New York, NY, USA, pp. 1227–1230, 2010.

[117] J. Zhao, C. Wu, and X. Lin, "Locality-aware streaming in hybrid p2p-cloud cdn systems," *Peer-to-Peer Networking and Applications*, vol. 8, no. 2, pp. 320–335, Mar 2015.

[118] O. Barais, J. Bourcier, Y.-D. Bromberg, and C. Dion, "Towards microservices architecture to transcode videos in the large at low costs," in *Proceedings of the IEEE International Conference on Telecommunications and Multimedia (TEMU)*, pp. 1–6, 2016.

[119] B. Khemka, A. A. Maciejewski, and H. J. Siegel, "A performance comparison of resource allocation policies in distributed computing environments with random failures," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pp. 1, June 2012.

[120] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, and S. Ali, "Representing task and machine heterogeneities for heterogeneous computing systems," *Journal of Applied Science and Engineering*, vol. 3, no. 3, pp. 195–207, Sept. 2000.

[121] A. M. Al-Qawasmeh, A. A. Maciejewski, R. G. Roberts, and H. J. Siegel, "Characterizing task-machine affinity in heterogeneous computing environments," in *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, pp. 34–44, May 2011.

[122] A. M. Al-Qawasmeh, A. A. Maciejewski, and H. J. Siegel, "Characterizing heterogeneous computing environments using singular value decomposition," in *Proceedings of the IEEE International Symposium on Parallel; Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pp. 1–9, Apr. 2010.

[123] M. A. Salehi, J. Smith, A. A. Maciejewski, H. J. Siegel, E. K. P. Chong, J. Apodaca, L. D. Briceno, T. Renner, V. Shestak, J. Ladd, A. Sutton, D. Janovy, S. Govindasamy, A. Alqudah, R. Dewri, and P. Prakash, "Stochastic-based robust dynamic resource allocation in heterogeneous computing system," *Parallel and Distributed Computing*, Nov. 2016.

[124] G. B. Berriman, G. Juve, E. Deelman, M. Regelson, and P. Plavchan, "The application of cloud computing to astronomy: A study of cost and performance," in *Proceedings of the 6th IEEE International Conference on-Science Workshops*, pp. 1–7, Oct. 2010.

[125] R. R. Expósito, G. L. Taboada, S. Ramos, J. Touriño, and R. Doallo, "General-purpose computation on gpus for high performance cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 12, pp. 1628–1642, May 2012.

[126] K. P. Puttaswamy, C. Kruegel, and B. Y. Zhao, "Silverline: toward data confidentiality in storage-intensive cloud applications," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, pp. 10, Oct. 2011.

[127] M.-J. Hsieh, C.-R. Chang, L.-Y. Ho, J.-J. Wu, and P. Liu, "Sqlmr: A scalable database management system for cloud computing," in *Proceedings of the 42nd International Conference on Parallel Processing*, pp. 315–324, Sept. 2011.

[128] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats," *IEEE Transactions on Multimedia*, vol. 2, no. 2, pp. 101–110, June 2000.

[129] P. Yin, M. Wu, and B. Liu, "Video transcoding by reducing spatial resolution," in *Proceedings of the International Conference on Image Processing*, vol. 1, pp. 972–975, Sept. 2000.

[130] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *Proceedings of the 24th IEEE international conference on advanced information networking and applications*, pp. 27–33, Apr. 2010.

[131] M. L. Puri and D. A. Ralescu, "Differentials of fuzzy functions," *Journal of Mathematical Analysis and Applications*, vol. 91, no. 2, pp. 552–558, 1983.

[132] M. Salehi and R. Buyya, "Adapting market-oriented scheduling policies for cloud computing," in *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing*, vol. 6081, pp. 351–362, Jan. 2010.

[133] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 15–28, 2007.

[134] "Quora," http://www.quora.com/, accessed: 2017.

[135] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.

[136] "tubefilter," http://http://www.tubefilter.com/, accessed: 2017.

[137] G. F. Jenks, "The data model concept in statistical mapping," *International yearbook of cartography*, vol. 7, no. 1, pp. 186–190, 1967.

[138] M. Hosseini, M. Amini Salehi, and R. Gottumukkala, "Enabling interactive video stream prioritization for public safety monitoring through effective batch scheduling," in *Proceedings of the 19th IEEE International Conference on High Performance Computing and Communications*, ser. HPCC '17, Dec. 2017.

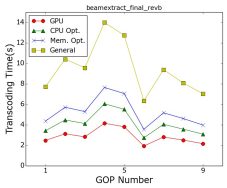
[139] M. Amini Salehi and X. Li, "HLSaaS: High-Level Live Video Streaming as a Service," in *Proceedings of the Workshop organized by Department of Energy, Stream2016*, Mar. 2016.

[140] Y. Lin and H. Shen, "Cloudfog: Leveraging fog to extend cloud gaming for thin-client mmog with high quality of service," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 28, no. 2, pp. 431–445, Feb. 2017.

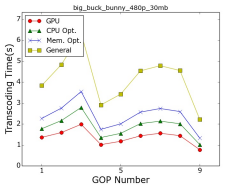
[141] X. Li, M. A. Salehi, M. Bayoumi, N. Tzeng, and R. Buyya, "Cost-efficient and robust on-demand video transcoding using heterogeneous cloud services," *IEEE Transactions on Parallel and Distributed Systems*, 2017.

APPENDIX A: Execution Time of Codec Videos Transcoding

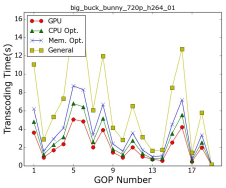
The following figures show the execution time of codec transcoding for all video in the benchmark on different VM types. We can observe that, in general, `GPU` VM provides a better execution time in comparison with other VM types. `General` VM provides the lowest performance as it includes less powerful processing units.



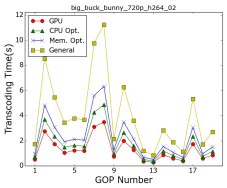
(a)



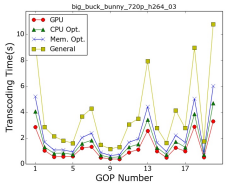
(b)



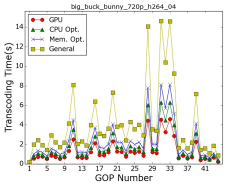
(c)



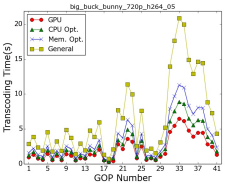
(d)



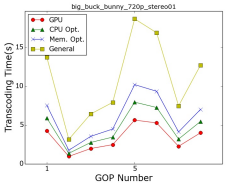
(e)



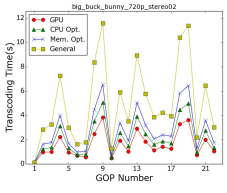
(f)



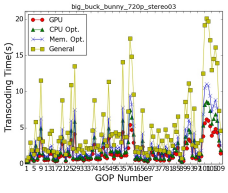
(g)



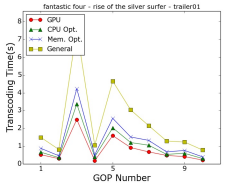
(h)



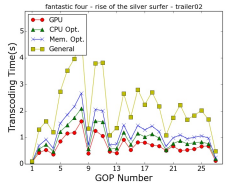
(i)



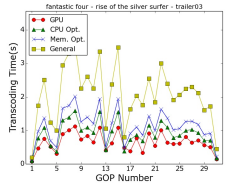
(j)



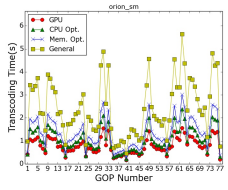
(k)



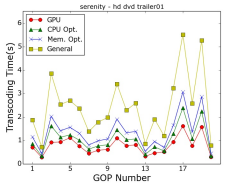
(l)



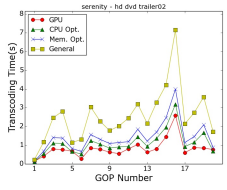
(m)



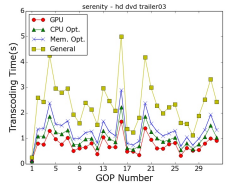
(n)



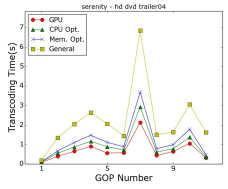
(o)



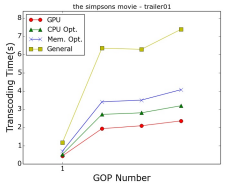
(p)



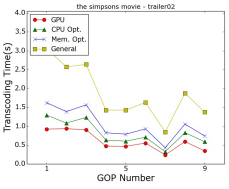
(q)



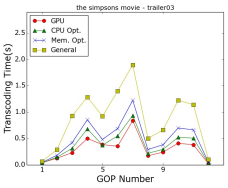
(r)



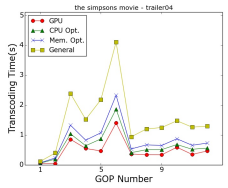
(s)



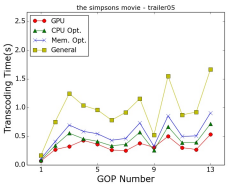
(t)



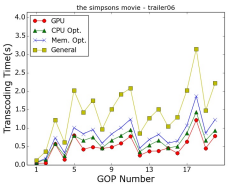
(u)



(v)



(w)



(x)

APPENDIX B: Comparison of all Methods

The following tables show a comparison of the incurred costs of using cloud resources when applying proposed partial methods and baseline methods on video streams in repository. Particularly, we show a comparison of the incurred cost of using cloud resources when applying two methods (Full Pre-Transcoding and Full Pre-Transcoding), which we didn't include in previous bar charts figures, because applying these two methods incurs very high costs compared to other methods, and the comparison becomes unclear.

Table B.1. Incurred costs of different partial pre-transcoding methods (in US Dollar) when the percentage of Frequently Accessed Video Streams varies in the repository.

% FAVs	FPT	FRT	VLT	GBH	VBH	GBTh
5%	3199	5136	2093	739	930	791
10%	3199	5157	2290	727	903	775
15%	3199	5290	2735	719	879	764
20%	3199	5810	3130	719	852	757
25%	3199	8735	5592	764	853	792
30%	3199	19040	13155	850	910	870

Table B.2. Incurred costs of different partial pre-transcoding methods when cloud storage and cloud CDN is used (in US Dollar)

% FAVs	FPT	FRT	VLT	GBH	VBH	GBTh
5%	12261	5047	2735	1147	1653	1259
10%	12261	5157	2725	1137	1633	1239
15%	12261	5290	1230	1130	1610	1230
20%	12261	5810	1243	1148	1600	1247
25%	12261	8735	1395	1312	1690	1403
30%	12261	19040	13157	1680	1951	1753

Table B.3. Incurred costs of different partial pre-transcoding methods when the variance of number of views for video streams increases (in US Dollar).

Variance	FPT	FRT	VLT	GBH	VBH	GBTh
1.16×10^7	3199	67236	2356	1721	4035	2172
2.59×10^7	3199	96303	1701	1410	2825	1565
4.61×10^7	3199	126280	1374	1203	2234	1261
7.2×10^7	3199	156621	1178	1056	1891	1079
10.37×10^7	3199	187144	1047	945	1672	958

Table B.4. Incurred costs of different partial pre-transcoding methods (in US Dollar) when the percentage of video streams with non-long-tail access pattern varies in the repository.

% videos with non-long tail access	FPT	FRT	VLT	GBH	VBH	GBTh
0%	3199	2338	1323	613	606	611
10%	3199	5363	2780	642	655	649
20%	3199	5618	2959	675	732	697
30%	3199	5810	3129	719	852	757
40%	3199	6082	3775	751	1033	802
50%	3199	6245	4163	774	1184	835
60%	3199	6323	4238	799	1258	871
70%	3199	6492	4441	823	1413	903
80%	3199	6548	4489	841	1468	928
90%	3199	6618	4528	856	1530	948
100%	3199	6661	4579	874	1576	973

Darwich, Mahmoud Bachelor of Science, Beirut Arab University, Spring, 2006; Master of Science, University of Louisiana at Lafayette, Fall, 2013; Doctor of Philosophy, University of Louisiana at Lafayette, Fall 2017.
Major: Computer Engineering
Title of Dissertation: Cost-Efficient Video On Demand (VOD) Streaming Using Cloud Services
Dissertation Director: Mohsen Amini Salehi and Magdy A. Bayoumi
Pages in Dissertation: 152; Words in Abstract: 312

ABSTRACT

Video streaming has become ubiquitous and pervasive in usage of the electronic displaying devices. Streaming becomes more challenging when dealing with an enormous number of video streams. Particularly, the challenges lie in streaming types, video transcoding, video storing, and video delivering to users with high satisfaction and low cost for video streaming providers. In this dissertation, we address the challenges and issues encountered in video streaming and cloud-based video streaming. Specifically, we study the impact factors on video transcoding in the cloud, and then we develop a model to trade-off between performance and cost of cloud. On the other hand, video streaming providers generally have to store several formats of the same video and stream the appropriate format based on the characteristics of the viewer's device. This approach, called pre-transcoding, incurs a significant cost to the stream providers that rely on cloud services. Furthermore, pre-transcoding is proven to be inefficient due to the long-tail access pattern to video streams. To reduce the incurred cost, we propose to pre-transcode only frequently-accessed videos (called *hot* videos) and partially pre-transcode others, depending on their hotness degree. Therefore, we need to measure video stream hotness. Accordingly, we first provide a model to measure the hotness of video streams. Then, we develop methods that operate based on the hotness measure and determine how to pre-transcode videos to minimize the cost of stream providers. The partial

pre-transcoding methods operate at different granularity levels to capture different patterns in accessing videos. Particularly, one of the methods operates faster but cannot partially pre-transcode videos with the non-long-tail access pattern. Experimental results show the efficacy of our proposed methods, specifically, when a video stream repository includes a high percentage of Frequently Accessed Video Streams and a high percentage of videos with the non-long-tail accesses pattern.

BIOGRAPHICAL SKETCH

Mahmoud K. Darwich received his Bachelor of Science degree in Spring 2006 in electrical engineering from Beirut Arab University, Lebanon. After working in a construction firm as an electrical engineer in Saudi Arabia for four years, he relocated to Lousiana, USA and pursued graduate studies at the University of Louisiana at Lafayette, receiving a Master of Science in computer engineering in 2013. Mahmoud K. Darwich completed the requirements for the Doctor of Philosophy in Computer Engineering from the University of Louisiana at Lafayette in Fall, 2017. Mahmoud K. Darwich currently holds the position of Assistant Professor of Information Technology at Navajo Technical University, New Mexico. His research interests include cloud computing, video transcoding, and VLSI circuits design.