Cost-Efficient Cloud-Based Video Streaming Through Measuring Hotness

Mahmoud Darwich[†‡], Mohsen Amini Salehi[‡], Ege Beyazit[†], Magdy Bayoumi[†]

[†] *The Center for Advanced Computer Studies,*
[‡] *High Performance Cloud Computing (HPCC) Laboratory*
School of Computing and Informatics
University of Louisiana at Lafayette, LA 70503, USA
Email: mdarwich@navajotech.edu, {amini, exb6143, mab0778}@louisiana.edu

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 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 the Frequently Accessed Video Streams and a high percentage of videos with the non-long-tail accesses pattern.

Keywords: Cloud Services, Video On Demand (VOD), Partial Pre-transcoding, Re-transcoding, Video Streaming, Video Stream Hotness.

Received 00 January 2009; revised 00 Month 2009

1. INTRODUCTION

Based on the Global Internet Phenomena Report [1], video streaming currently constitutes around 64% of the U.S. Internet traffic and is rocketing to 80% of the whole Internet traffic by 2019 [2]. Viewers stream videos on a variety of devices with different architectural characteristics, from large screen TVs and desktops to tablets and smart-phones. Video contents, either in form of Video On Demand (VOD) (*e.g.,* YouTube[1] or Netflix[2]) or live-streaming (*e.g.,* Livestream[3]), need to be converted (*i.e.,* transcoded) based on the characteristics of viewers devices (*e.g.,* in terms of frame rate, resolution, and available network bandwidth) [3].

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

To make the video streams readily available for viewers video stream providers commonly carry out the transcoding operation in an off-line 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 [4] and store them in their repositories. To overcome the storage and computational demands of transcoding, video stream providers extensively utilize cloud services [5]. Making use of cloud services imposes a significant cost overhead to video stream providers [6, 7], hence, they are actively seeking solutions to reduce their incurred cost of using clouds services [8, 9].

To reduce the incurred cost, video stream providers need to be aware of access patterns to their video streams. Recent studies show that access-

ing video streams in a repository follows a long-tail distribution [10]. 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 are rarely accessed. Several research works have been undertaken (*e.g.,* [6, 7]) 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 generally more expensive than the storage cost [11]. 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 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 *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. In addition, 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 paper, 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 video stream repository management method 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 research must be applied on 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, hence, reduce the execution time overhead of repository management methods.

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

- Providing a formal definition for hot videos and a method to measure the hotness of video streams in the repository.
- Proposing video stream repository management methods to reduce the incurred cost of video stream providers in using cloud services. The methods choose to 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 video stream repository management methods.
- Analyzing the impact of video stream repository management methods on video streams with different access patterns.

Experimental results demonstrate that our proposed video streams repository management methods can reduce the incurred cost of using cloud services significantly. Thus, the research outcome of this paper 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 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 a repository to minimize the execution time of our methods.

The rest of the paper is organized as follows: section 2 provides some background on video streaming and transcoding. Section 3 presents the system model. Proposed algorithms are described in section 5. Section 4 presents the quantification of video stream hotness. In section 6, a clustering method is proposed. Experiment setup is detailed in section 7. Experimental results are discussed in section 8, related works are presented in section 9 and finally section 10 concludes the paper.

2. BACKGROUND

2.1. Video Stream Structure

A video stream is composed of a set of sequences as shown in Figure 1. The first block of a sequence is called a sequence header that contains meta-data about that sequence and is followed by several Group Of Pictures (GOP). Each GOP is constructed of a GOP header followed by several frame types, starting with I (intra) frame, followed by a P (predicted) and B (bi-directional) frames. Each frame is further comprised of slices that are formed from macroblocks (MB) [12].

As each GOP can be processed independently, video transcoding operation is commonly performed at the GOP level [12] *i.e.,* each GOP is considered as a unit for re-transcoding or pre-transcoding.

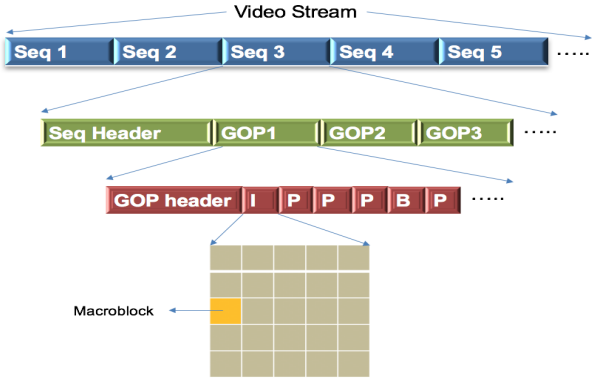


FIGURE 1: Video Stream Structure

2.2. Video Transcoding

Originally, Video streams are captured with a particular format, spatial resolution, frame rate, and bit rate. Before streaming the videos, they should be adjusted according to the viewer's device resolution, frame rate, and video codec. [3]. These adjustments are generally termed video transcoding and are explained as follows:

- *Bit Rate Adjustment:* Video stream providers usually change the bit rate of video streams to ensure pleasant streaming [13]. Dynamic bit rate adjustment of video streams is also known as Adaptive video streaming [14].
- *Spatial Resolution Reduction:* This type of transcoding changes the dimensional size of the video to fit the screen size of viewers' devices. There are algorithms which can be applied to maintain the quality when transcoding videos [15].
- *Temporal Resolution Reduction:* Temporal resolution reduction is used when the viewer's device supports low frame rate. This can be done by dropping some frames of the video [16].
- *Video Compression Standard Conversion:* Applying compression standards on videos to reduce their size so as to be streamed smoothly and stored in small size. There are several compression techniques—from MPEG2 [17], to H.264 [18], and to the most recent one, HEVC [19].

2.3. Cloud Services for Video Stream Transcoding

Cloud providers offer various services in an on-demand manner and charge their users in a pay-as-you-go manner [11]. A cloud-based video stream provider generally utilizes the following cloud services:

- Computational Services: In clouds, computational services are generally provided through Virtual Machines (VMs) and users are charged based on the amount of time using them (generally, on an hourly basis).
- Storage Services: Users of storage services are charged based the volume of their stored data on the cloud.
- Content Delivery Network (CDN) services: CDN is a technology that reduces the delay to access static content types, including video streams, through the Internet [20]. CDN replicates the content (*e.g.,* video content) in different geographical areas to minimize the network travel time of content to users [20, 21]. Generally, video stream providers utilize CDNs to deliver video streams to viewers with the minimum delay [22, 23].
  Typically, cloud-based video streaming platforms consist of storage servers for storing videos, transcoding servers for tasks processing, and edge servers (*i.e.,* CDNs) for delivering transcoded video streams to viewers [24, 25]. In fact, CDN services are required, in addition to storage services to perform pre-transcoded cloud-based video streaming. However, the CDN cost is not applied when we provide on-demand video transcoding service.

Amazon Web Services[4] is a major cloud provider and offers all the foregoing cloud services in a reliable manner [26]. Although this study is independent of AWS technology and can work on any of cloud provider, we consider AWS services, charging model, and costs for our evaluations.

Amazon Elastic Compute Cloud (Amazon EC2) provides computational services in form of VMs. It offers various types of VMs to cover different computational demands. General purpose `t2-small` VM type is the most common service used for different type of processing and we utilize this VM type for our evaluations as well. The hourly cost of `t2-small` VM is $0.026. Amazon Simple Storage Service (Amazon S3) is the storage service of Amazon cloud. Amazon S3 costs $0.03 for each Gigabyte of stored data in a month. Amazon also offers the CDN service (called CloudFront[5]). It delivers the content to users (*i.e.,* viewers) through a worldwide network of data centers with the minimum delay [27]. Amazon charges $0.085 for each Gigabyte of data uses CloudFront for the first 10 TB per month.

3. SYSTEM MODEL

For the sake of clarity, the symbols that are used in this paper are summarized in Table 1.

3.1. 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

[4] https://aws.amazon.com/
[5] https://aws.amazon.com/cloudfront/

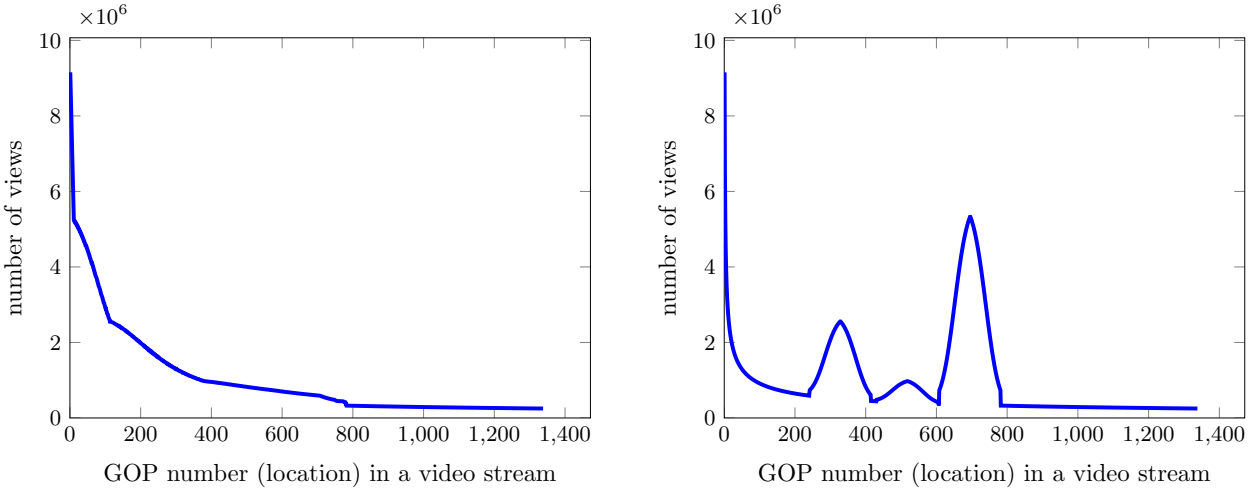


FIGURE 2: different shapes of non-long access for different video streams

the viewer within a time period. However, it does not determine if the requested video stream is watched to end of it or not. In fact, recent studies (*e.g.,* [28]) reveal that, in a video stream, the beginning GOPs are watched more frequently than the rest of the video stream. Miranda *et al.* [28] 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 [29] model. Based on this model, for a video stream with video access rate v_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 1. In this equation, α is called the power coefficient and its value is 0.1.

$$\varepsilon_{ij} = v_i \cdot j^{-\alpha} \tag{1}$$

Although 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 [30]. 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 to have called non-long-tail access pattern.

Figure 2 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 2a, we notice that the GOP access rate consistently decreases for GOPs later in the stream whereas, Sub-figure 2b demonstrates that GOP access rate can be significantly different regardless of the position of the GOP in the video stream.

Symbols	Description
m_i	Total number of GOPs in video i
G_{ij}	GOP number j of video i
v_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}

TABLE 1: Symbols used in proposed methods

3.2. Repository of Video Streams

Video stream providers keep enormous repositories of video streams [31]. For instance, the size of the repository of YouTube is estimated to reach over one billion GB [32]. In fact, storing video repositories in the cloud has become one of the main costs of video stream providers [33]. One main reason to maintain a large size repository is to store several 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 [28]. This means that few video streams in the repository are watched very frequently (these videos are known as *hot* videos), while the rest of video streams which are rarely watched. It has been reported that in YouTube provider only 5% of videos are frequently accessed and the rest are rarely watched [34].

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 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 the pre-transcoding or re-transcoding the video stream.

4. QUANTIFYING HOTNESS OF VIDEO STREAMS

4.1. 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 on the cloud. Then, we introduce our method to quantify video stream hotness in Subsection 4.4.

4.2. Cost of Pre-transcoding

The cost of the 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 [6]) 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 2.

$$C_{S_i} = S_i \cdot P_S \tag{2}$$

Another cloud service that is commonly used for storing and delivering video streams is CDN (*e.g.,* CloudFront in Amazon Cloud [7]). 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 3.

$$C_{S_i} = S_i \cdot (P_S + P_{CDN}) \tag{3}$$

4.3. 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.

$$C_{T_i} = P_T \cdot \tau_i \tag{4}$$

It is noteworthy that Equation 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) can be achieved based on the past[8] 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}.

4.4. 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 2 and 4. Let S_{ij} the size of GOP G_{ij}, then the pre-transcoding cost of G_{ij} is calculated based on Equation 5.

$$C_{S_{ij}} = S_{ij} \cdot P_S \tag{5}$$

Similarly, the estimated cost of re-transcoding G_{ij}, denoted $C_{T_{ij}}$, is calculated based on Equation 6.

$$C_{T_{ij}} = P_T \cdot \tau_{ij} \tag{6}$$

Using these concepts, we can introduce the GOP transcoding cost ratio, denoted R_{ij}, based on Equation 7.

$$R_{ij} = \frac{C_{S_{ij}}}{\varepsilon_{ij} \cdot C_{T_{ij}}} \tag{7}$$

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 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.,* $R_{ij} \leq$

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

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

[8] 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.

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 8.

$$h_{ij} = \begin{cases} 1 & R_{ij} \leq 1 \\ 0 & R_{ij} > 1 \end{cases} \quad (8)$$

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 3.1), 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 9.

$$H_i = \frac{\sum_{j=1}^{m} h_{ij}}{m} \quad (9)$$

Based on Equation 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.

5. VIDEO STREAM REPOSITORY MANAGEMENT METHODS

5.1. 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 metadata of video streams in the 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 5.2) operates at the video stream level whereas the other one (explained in Section 5.3) is more granular and operates at the GOP level.

5.2. 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 1.

This method calculates pre-transcoding and re-transcoding costs for each GOP of a video stream based Equations 5 and 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 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 of 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 1).

Algorithm 1: 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

1 **For each** video $V_i \in$ repository
2 **For each GOP** $G_{ij} \in V_i$
3 Calculate pre-transcoding cost: $C_{S_{ij}} \leftarrow S_{ij} \cdot P_S$
4 Calculate re-transcoding cost: $C_{T_{ij}} \leftarrow \tau_{ij} \cdot P_T$
5 Estimated No. of views for G_{ij}: $\varepsilon_{ij} \leftarrow \upsilon_i \cdot G_{ij}^{-\alpha}$
6 Calculate cost ratio for G_{ij}: $R_{ij} \leftarrow \frac{C_{S_{ij}}}{\varepsilon \cdot C_{T_{ij}}}$
7 **if** $R_{ij} \leq 1$ **then**
8 $h_{ij} \leftarrow 1$
9 Hotness measure for V_i: $H_i \leftarrow \frac{\sum_{j=1}^{m} h_{ij}}{m}$
10 No. of GOPs for pre-transcoding: $\chi_i \leftarrow \lceil H_i \cdot m \rceil$
11 Pre-transcode GOP_{i1} to $GOP_{i\chi}$ of V_i

The time complexity of Algorithm 1 is $T(n, m) = c \cdot (n \cdot m)$ where c is a constant, n is the number of videos in the repository and m is the number of GOPs in each video. Hence, the time complexity of the algorithm 1 is $O(nm)$. Similarly, the space complexity of the

algorithm is $O(nm)$.

5.3. 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 time 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 1, the method has to be performed for all video streams in the repository.

The pseudo-code for this method is shown in Algorithm 2. In this method, the transcoding cost-ratio for each GOP G_{ij} is calculated based on Equation 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 needs to be pre-transcoded. Otherwise, G_{ij} is re-transcoded.

It is worth noting that, in Algorithm 1, 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 1, 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 2 requires views information for each GOP and we need to maintain meta-data information for all GOPs for each video stream in the repository.

It is worth noting that both the time and space complexity of Algorithm 2 is $O(nm)$.

Algorithm 2: 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

1 **For each** video $V_i \in$ repository
2 **For each** GOP $G_{ij} \in V_i$
3 Calculate pre-transcoding cost: $C_{S_{ij}} \leftarrow S_{ij} \cdot P_S$
4 Calculate re-transcoding cost:$C_{T_{ij}} \leftarrow P_T \cdot \tau_{ij}$
5 Calculate cost ratio: $R_{ii} \leftarrow \frac{C_{S_{ij}}}{\psi_{ij} \cdot C_{T_{ij}}}$
6 **if** $R_{ij} \leq 1$ **then**
7 $h_{ij} = 1$
8 pre-transcode G_{ij}
9 **else**
10 $h_{ij} = 0$
11 re-transcode G_{ij}

6. REDUCING THE EXECUTION TIME OVERHEAD OF THE PARTIAL PRE-TRANSCODING METHOD

The partial pre-transcoding methods mentioned in the previous section 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.

Our proposed mechanism works based on clustering of 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 [35], which is a one-dimensional clustering (grouping) method, to group video streams that require similar kind of re-transcoding or pre-transcoding operation. Jenks Natural Breaks is an iterative algorithm aims to maximize the Goodness Of Variance (GOV) fit and is defined based on Equation 10.

$$GOV = \frac{SDV - SDC}{SDV} \tag{10}$$

where SDV is defined as the sum of squared deviations of views of video streams in a repository and is calculated based on Equation 11. To maximize GOV, we only need to minimize the sum of squared deviations for each cluster.

$$SDV = \sum_{i=1}^{N} (\upsilon_i - \overline{\upsilon_i})^2 \tag{11}$$

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 $\overline{v_i} = \frac{\sum_{i=1}^{N} v_i}{N}$. SDC is defined as the sum of squared deviations of video stream views across clusters and is calculated based on Equation 12.

$$SDC = \sum_{\kappa=1}^{c} (V_\kappa - \overline{V_\kappa})^2 \quad (12)$$

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 number 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 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.

7. EXPERIMENTAL SETUP

7.1. Synthesizing Video Streams

A video stream providers generally has a large video stream repository[9]. 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[10].

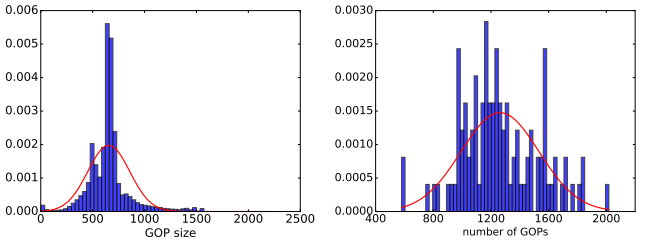
To accurately synthesize video streams, we need to know the distribution and characteristics of video streams in terms of GOP size, GOP transcoding time, and number of GOPs in each video. In order to extract this meta-data, we transcoded the video streams on Amazon EC2[11] and analyzed them.

The result of our analysis, shown in Figure 3a and 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 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). As such, we can derive a linear equation to estimate transcoding time of a GOP based on its size.

[9]YouTube repository is estimated to be one billion GB [32].
[10]The videos can be downloaded from: https://goo.gl/TE5iJ5
[11]https://aws.amazon.com/ec2

	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

TABLE 2: Meta-data of GOP size and number of GOPs extracted from video streams of our repository.



(a) GOP size histogram (b) GOPs of video histogram

FIGURE 3: Distribution of GOP size and numbers of GOPs in video streams of our repository.

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

7.2. 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.* [10], 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 2a 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 2b depicts a video stream with four peak views, each one has a different height and length.

We determine the height of each peak view point by choosing a random number between minimum and maximum access values of that video. The length of

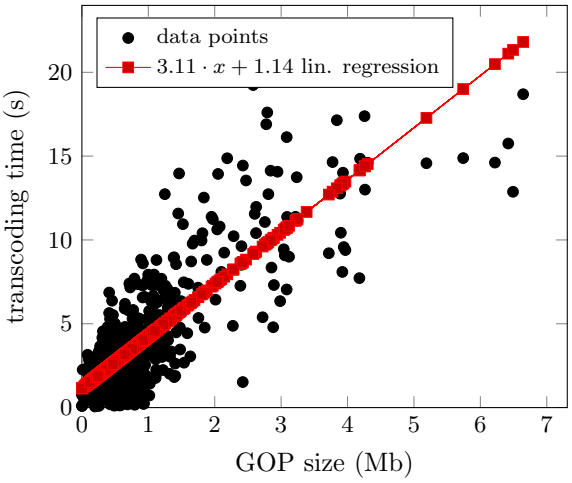


FIGURE 4: Linear regression analysis of GOP size and its transcoding time.

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

TABLE 3: Incurred cost of computation, storage, and CDN in AWS cloud. All costs are in US dollar.

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 $(300, 100)$.

7.3. 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 3.

7.4. 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 works in the literature. For the sake accuracy of our evaluations, all experiments in this research 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 prior to be 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.* [36] 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 9.

GOP-Based Threshold (GBTh): We proposed this method in our earlier work [37]. 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.

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

TABLE 4: Percentage of Frequently Accessed Video Streams (FAVs) in the synthesized repositories by varying the Shape parameter (α) in the Weibull distribution.

8. PERFORMANCE EVALUATIONS

8.1. 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 varying 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 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. The average number of views in all synthesized repositories is 1.99.

Figure 5a shows the total cost incurred by different methods, explained in Subsection 7.4. The vertical axis, in Figure 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 Appendix A section to see the detailed comparison with these methods. In Figure 5a, we observe that as the percentage of Frequently Accessed Video 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 [36] 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 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 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 a repository is Frequently Accessed Video Streams , GBH reduces the total incurred 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.

8.2. 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, our aim is to evaluate the behavior of the proposed partial pre-transcoding methods under different long-tail patterns. For that purpose, we alter the variance of the 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 the 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 6 shows the results of the experiment. In this figure, the horizontal axis shows the variance of the 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 streams are 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.

8.3. 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 3.1). The percentage

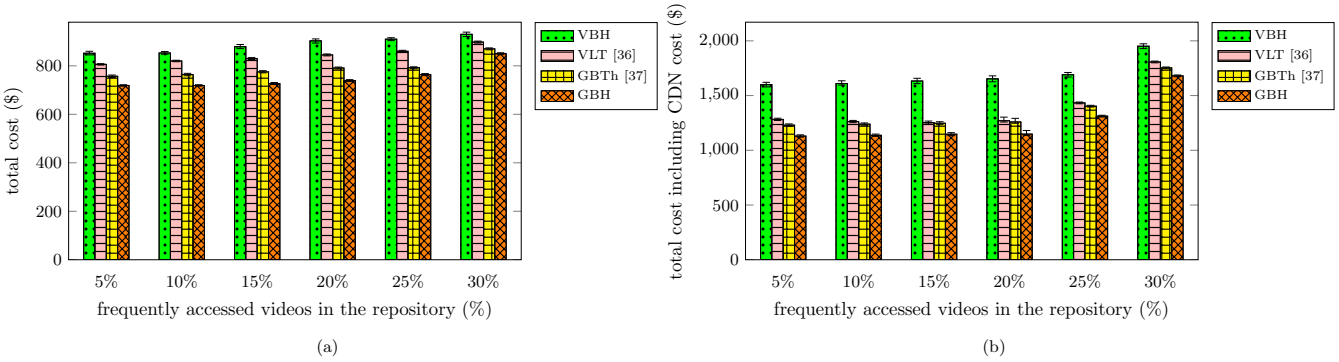


FIGURE 5: (a) Incurred costs of different partial pre-transcoding methods (in US Dollar) when the percentage of Frequently Accessed Video Streams varies in the repository. (b) Incurred costs of different partial pre-transcoding methods when cloud storage and cloud CDN is used (in US Dollar). Horizontal axis shows the percentage of Frequently Accessed Video Streams .

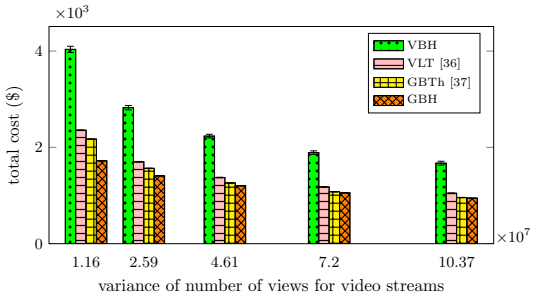


FIGURE 6: Incurred costs of different partial pre-transcoding methods when the variance of number of views for video streams increases (in US Dollar).

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 7.2) and measured the incurred costs resulted from different partial pre-transcoding methods. Figure 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 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.

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 the 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 streams that have non-long-tail access pattern. We can conclude that when the percentage of video streams with the non-long-tail access pattern is high and dominates the whole repository, it is worthwhile to investigate the cost-efficiency for each GOP.

8.4. Execution Time Overhead of 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 6) 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 8 shows the results of the experiment. The

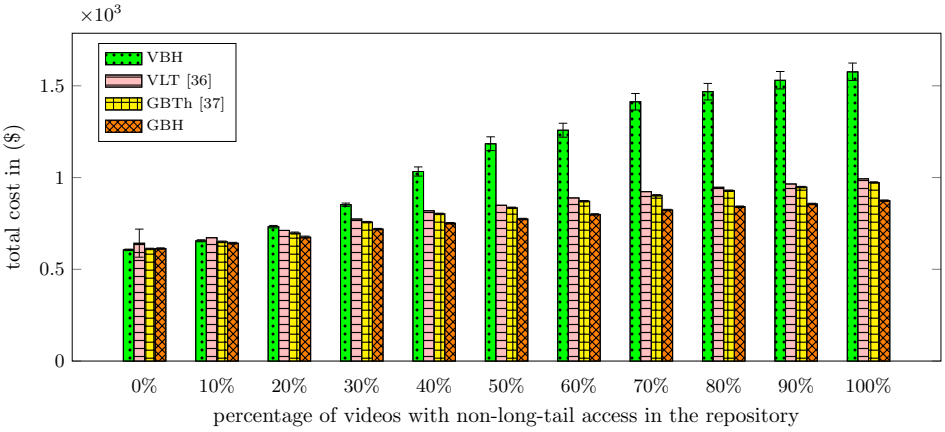


FIGURE 7: Incurred costs of different partial pre-transcoding methods (in US Dollar) when the percentage of video streams with non-long-tail access pattern varies in the repository.

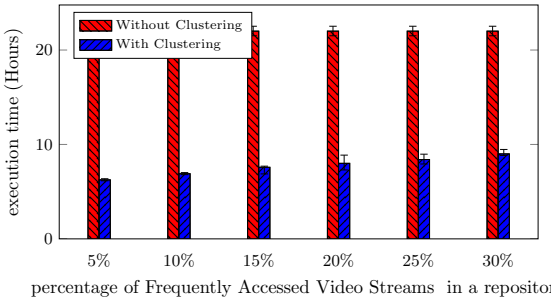


FIGURE 8: Execution time overhead of partial pre-transcoding methods (in hours) with and without clustering video streams in the repository.

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 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 the distribution of the number of views for video streams in the repository follows a long-tail distribution. That is, the frequently accessed videos are located at the beginning of the curve and the rest of video streams, which are rarely accessed, are located in the tail of the curve. When the percentage of the frequently accessed videos in the repository is low, the number of partially hot video streams is low too and vice versa. Accordingly, when the percentage of the frequently accessed video streams increases, the percentage of the rarely accessed video streams decreases. GBH method is particularly applied to the cluster of partially hot video streams, hence, its execution time increases by increasing the percentage of frequently accessed video streams.

9. RELATED WORK

Gao *et al.* [38] 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 transcoding the remaining video contents upon request.They demonstrated that their method reduces 30% of the operational cost in comparison to storing the whole videos. In another research, Gao *et al.* [39] proposed a method based on linear integer programming to determine the optimal partial pre-transcoding. Our research is different with these works in the sense that we provide a method to measure the hotness of video streams. In addition, we propose, methods that can cover video streams that do not follow a long-tail access pattern.

Zhao *et al.* [40, 41] proposed a trade-off between computation and storage costs and to minimize the cost for 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 a reduction in the incurred cost when compared to storing all versions of videos.

Jokhio *et al.* [36] 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.

Krishnappa *et al.* [25] proposed transcoding policies that transcode video segments that are requested by users. In order to maintain the quality of videos in terms of startup 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 resources with high accuracy.

In our earlier work [37], we proposed a method to partially pre-transcode a video stream with the long-tail access pattern. Our method finds the first GOP (called threshold GOP) that has its transcoding cost ratio greater than 1; GOPs located before the threshold GOP are pre-transcoded and those located after the threshold GOP are re-transcoded. The method reduces the cost of video transcoding using clouds services. Our method is efficient for a repository that contains video streams with the long-tail access pattern.

Li *et al.* [42] proposed Cloud-based Video Streaming Services (CVS2) architecture. It includes a QoS-aware scheduling component that maps transcoding tasks to the Virtual Machines (VMs) . To maintain robustness in the presence of varying streaming requests, the architecture includes a cost-efficient VM Provisioner component. Simulation results obtained under diverse workload conditions demonstrate that CVS2 architecture can maintain a robust QoS for viewers while reducing the incurred cost of the streaming service provider by up to 85%.

Karolewicz and Beben [43] proposed provisioning methods for VoD service providers offering adaptive video streaming in a cloud environment. Their methods minimized costs paid by VoD provider. They formulated provisioning problem and proposed optimization and heuristic methods that determine the best tradeoff between content storage and transcoding. The results of performance evaluation confirmed that proposed methods can significantly reduce costs paid by VoD providers.

Atish *et al.* [44] developed cost metrics that allow to compare storage vs. compute costs and suggest when a transcoding on-the-fly solution can be cost-effective. They also analyzed how such a solution can be deployed in a practical storage system using access pattern information or a variant of the ski-rent online algorithm when such information is not available.

A joint computing-plus-communication optimization framework exploiting virtualization technologies was proposed to address the typical scenario of multimedia data processing with computationally intensive tasks and exchange of a big volume of data. The proposed framework not only ensures users the Quality of Service (through Service Level Agreements), but also achieves maximum energy saving and attains green cloud computing goals in a fully distributed fashion [45].

A paradigm is based on real-time energy-efficient management of the distributed resources available at both mobile devices and Internet-connected data centers was proposed by Baccarelli *et al.* in [46]. The proposed paradigm overcomes the resources limitation imposed by mobile/wireless devices to process the big data stream.

Hosseini *et al.* [47] proposed a stream-priority aware resource allocation mechanism to enable interactive video prioritization without a major impact on the flow of nonprioritized video streams. Their mechanism includes a method to select appropriate tasks from the arriving ones and a method to map the selected task to the appropriate video server. Their results showed that the percentage of normal and prioritized video streaming tasks that have completed on-time is improved when compared with baseline scheduling methods.

Amini Salehi *et al.* [48] proposed a stochastic robustness measure to facilitate resource allocation decisions in a dynamic environment where tasks are subject to individual hard deadlines and each task requires some input data to start execution. they also proposed methods to determine the stochastic completion times of tasks in the presence of the task dropping. Moreover, they designed novel resource allocation techniques that work in immediate and batch modes, with the goal of maximizing the number of tasks that meet their individual deadlines. their results demonstrated the suitability of the proposed technique in a dynamic heterogeneous computing system.

10. CONCLUSION AND FUTURE WORK

The goal of this research is 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 need to measure the hotness of the video streams. Accordingly, in this paper, we proposed a method to quantify the hotness of video streams. The hotness measure then used to develop repository management methods that are executed periodically and determine 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 demonstrate 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 the 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 efficient when the repository includes a high percentage of videos with the 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%.

In future, we plan to work on the prediction of the number of views for a video stream by applying machine learning techniques. The prediction can operate based on the number of accesses to the video streams in the past periods. Another future direction of this research is to perform dynamic video stream summarization based on the access pattern to GOPs in a video stream.

ACKNOWLEDGMENTS

We are thankful for the comments of Dr. Xiangbo Li who is an expert in the video streaming industry. Also, we are thankful the comments of anonymous reviewers of the paper. This research was supported by the Louisiana Board of Regents under grant number LEQSF(2016-19)-RD-A-25. This is a substantially extended version of a paper presented in the Proceedings of the 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (IEEE Mobile Cloud '17) [37].

REFERENCES

[1] G. I. P. Report. `http://www.sandvine.com/trends/global-internetphenomena/`. October, 2016.

[2] C. V. N. Index, Forecast and methodology, 2016-2021. `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html`. June, 2017.

[3] Ahmad, I., Wei, X., Sun, Y., and Zhang, Y.-Q. (2005) Video transcoding: an overview of various techniques and research issues. *IEEE Transactions on multimedia*, **7**, 793–804.

[4] Netflix. `http://techblog.netflix.com/2012/12/videos-of-netflix-talks-at-aws-reinvent.html`. December, 2012.

[5] Li, X., Amini Salehi, M., and Bayoumi, M. (2015) Cloud-based video streaming for energy-and compute-limited thin clients. *Stream 2015 Workshop at Indiana University*.

[6] Li, X., Amini Salehi, M., and Bayoumi, M. (2016) Vlsc: Video live streaming using cloud services. *Proceedings of 5th IEEE International Conference on Big Data and Cloud Computing (BDCloud '16)*, Atlanta, GA, 8-10 October, pp. 595–600. IEEE, USA.

[7] Li, X., Amini Salehi, M., Bayoumi, M., and Buyya, R. (2016) Cvss: A cost-efficient and qos-aware video streaming using cloud services. *Proceeding of 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, Cartagena, Colombia, 16-19 May, pp. 106–115. IEEE, USA.

[8] Li, H., Zhong, L., Liu, J., Li, B., and Xu, K. (2011) Cost-effective partial migration of vod services to content clouds. *Proceedings of 2011 IEEE International Conference on Cloud Computing (CLOUD)*, Cartagena, Colombia, 16-19 May, pp. 203–210. IEEE, USA.

[9] Amini Salehi, M. and Buyya, R. (2010) Adapting market-oriented scheduling policies for cloud computing. *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing*, Busan, South Korea, 21-23 May, ICA3PP '10, **6081**, pp. 351–362. Springer, Berlin, Heidelberg.

[10] Sharma, N., Krishnappa, D. K., Irwin, D., Zink, M., and Shenoy, P. (2013) Greencache: Augmenting off-the-grid cellular towers with multimedia caches. *Proceedings of the 4th ACM Multimedia Systems Conference*, Oslo, Norway, February 28 - March 01, pp. 271–280. ACM, New York, NY.

[11] Amazon. `https://aws.amazon.com/ec2/pricing/on-demand/`. August, 2017.

[12] Jokhio, F., Deneke, T., Lafond, S., and Lilius, J. (2011) Analysis of video segmentation for spatial resolution reduction video transcoding. *Proceedings of the International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS),*, Chiang Mai, Thailand, Thailand, 7-9 December, pp. 1–6. IEEE, USA.

[13] Werner, O. (1999) Requantization for transcoding of MPEG-2 intraframes. *IEEE Transactions on Image Processing*, **8**, 179–191.

[14] Jiang, J., Sekar, V., and Zhang, H. (2014) Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. *Proceedings of the 8th International Conference on emerging Networking Experiments and Technologies*, February, pp. 97–108. IEEE Press Piscataway, NJ.

[15] Bjork, N. and Christopoulos, C. (1998) Transcoder architectures for video coding. *IEEE Transactions on Consumer Electronics*, **44**, 88–98.

[16] Goel, S., Ismail, Y., and Bayoumi, M. (2012) High-speed motion estimation architecture for real-time video transmission. *The Computer Journal*, **55**, 35–46.

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

[18] Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A. (2003) Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, **13**, 560–576.

[19] Sullivan, G. J., Ohm, J.-R., Han, W.-J., and Wiegand, T. (2012) Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, **22**, 1649–1668.

[20] Vakali, A. and Pallis, G. (2003) Content delivery networks: Status and trends. *IEEE Internet Computing*, **7**, 68–74.

[21] Saroiu, S., Gummadi, K. P., Dunn, R. J., Gribble, S. D., and Levy, H. M. (2002) An analysis of internet content delivery systems. *ACM SIGOPS Operating Systems Review*, **36**, 315–327.

[22] Akamai. `http://www.akamai.com/`. August, 2017.

[23] Level3. `http://www.level3.com/`. August, 2017.

[24] Zhuang, Z. and Guo, C. (2012) Building cloud-ready video transcoding system for content delivery networks (cdns). *Proceeding of Global Communications Conference (GLOBECOM)*, Anaheim, CA, 3-7 December, pp. 2048–2053. IEEE, USA.

[25] Krishnappa, D. K., Zink, M., and Sitaraman, R. K. (2015) Optimizing the video transcoding workflow in content delivery networks. *Proceedings of the 6th ACM Multimedia Systems Conference*, Portland, Oregon, 18 - 20 March, pp. 37–48. ACM, New York, NY.

[26] Amazon. `http://aws.amazon.com/`. August, 2017.

[27] CloudFront. `http://docs.aws.amazon.com/`. August, 2017.

[28] Miranda, L. C., Santos, R. L., and Laender, A. H. (2013) Characterizing video access patterns in mainstream media portals. *Proceedings of the 22nd International Conference on World Wide Web*, Rio de Janeiro, Brazil, 13-27 May, pp. 1085–1092. ACM, New York, NY.

[29] Newman, M. E. (2005) Power laws, pareto distributions and zipf's law. *Contemporary physics*, **46**, 323–351.

[30] Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.-Y., and Moon, S. (2007) I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, San Diego, California, 24-26 October, pp. 1–14. ACM, New York, NY.

[31] Gill, P., Arlitt, M., Li, Z., and Mahanti, A. (2007) Youtube traffic characterization: a view from the edge. *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, San Diego, California, 24-26 October, pp. 15–28. ACM, New York, NY.

[32] Quora. `http://www.quora.com/`. August 2016.

[33] Zhang, Q., Cheng, L., and Boutaba, R. (2010) Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, **1**, 7–18.

[34] tubefilter. `http://http://www.tubefilter.com/`. 2016.

[35] Jenks, G. F. (1967) The data model concept in statistical mapping. *International yearbook of cartography*, **7**, 186–190.

[36] Jokhio, F., Ashraf, A., Lafond, S., and Lilius, J. (2013) A computation and storage trade-off strategy for cost-efficient video transcoding in the cloud. *Proceeding of 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, Santander, Spain, 4-6 September, pp. 365–372. IEEE, USA.

[37] Darwich, M., Beyazit, E., Amini Salehi, M., and Bayoumi, M. (2017) Cost efficient repository management for cloud-based on-demand video streaming. *Proceedings of 5th IEEE Internationa Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, San Francisco, CA, 6-8 April, pp. 1–6. IEEE, USA.

[38] Gao, G., Zhang, W., Wen, Y., Wang, Z., and Zhu, W. (2015) Towards cost-efficient video transcoding in media cloud: Insights learned from user viewing patterns. *IEEE Transactions on Multimedia*, **17**, 1286–1296.

[39] Gao, G., Zhang, W., Wen, Y., Wang, Z., Zhu, W., and Tan, Y. P. (2014) Cost optimal video transcoding in media cloud: Insights from user viewing pattern. *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, Chengdu, China, 14-18 July, pp. 1–6. IEEE, USA.

[40] Zhao, H., Zheng, Q., Zhang, W., Du, B., and Chen, Y. (2015) A version-aware computation and storage trade-off strategy for multi-version vod systems in the cloud. *Proceeding of IEEE Symposium onComputers and Communication (ISCC)*, Larnaca, Cyprus, 6-9 July, pp. 943–948. IEEE, USA.

[41] Zhao, H., Zheng, Q., Zhang, W., Du, B., and Li, H. (2017) A segment-based storage and transcoding trade-off strategy for multi-version vod systems in the cloud. *IEEE Transactions on Multimedia*, **19**, 149–159.

[42] Li, X., Amini Salehi, M., Bayoumi, M., Tzeng, N., and Buyya, R. (2018) Cost-efficient and robust on-demand video transcoding using heterogeneous cloud services. *IEEE Transactions on Parallel and Distributed Systems*, **29**, 556 – 571.

[43] Karolewicz, K. and Bben, A. (2017) Cloud-based adaptive video streaming: Content storage vs. transcoding optimization methods. *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*, Heraklion, Greece, 3-6 July, pp. 523–528. IEEE, USA.

[44] Kathpal, A., Kulkarni, M., and Bakre, A. (2012) Analyzing compute vs. storage tradeoff for video-aware storage efficiency. *Presented as part of the 4th USENIX Workshop on Hot Topics in Storage and File Systems*, Boston, MA, 13-14 June. USENIX, Berkeley, CA.

[45] Shojafar, M., Canali, C., Lancellotti, R., and Abawajy, J. (2016) Adaptive computing-plus-communication optimization framework for multimedia processing in cloud systems. *IEEE Transactions on Cloud Computing*, **PP**, 1–1.

[46] Baccarelli, E., Cordeschi, N., Mei, A., Panella, M., Shojafar, M., and Stefa, J. (2016) Energy-efficient dynamic traffic offloading and reconfiguration of networked data centers for big data stream mobile computing: review, challenges, and a case study. *IEEE Network*, **30**, 54–61.

[47] Hosseini, M., Amini Salehi, M., and Gottumukkala, R. (2017) Enabling interactive video stream prioritization for public safety monitoring through effective batch scheduling. *Proceedings of the 19th IEEE International Conference on High Performance Computing and Communications*, Bangkok, Thailand, 18-20 December, pp. 474 – 481. IEEE, USA.

[48] Amini Salehi, M., Smith, J., Maciejewski, A. A., Siegel, H. J., Chong, E. K., Apodaca, J., Briceño, L. D., Renner, T., Shestak, V., Ladd, J., et al. (2016) Stochastic-based robust dynamic resource allocation for independent tasks in a heterogeneous computing system. *Journal of Parallel and Distributed Computing*, **97**, 96–111.

[49] Li, X., Joshi, Y., Darwich, M., Landreneau, B., Amini Salehi, M., and Bayoumi, M. (2017) Performance analysis and modeling of video transcoding using heterogeneous cloud services. *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–12.

[50] Belkin, M. and Niyogi, P. (2001) Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, British Columbia, Canada, 03 - 08 December, pp. 585–591. ACM, MIT Press Cambridge, MA.

APPENDIX A.

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 a 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 clear.

% 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 A.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%	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 A.2: Incurred costs of different partial pre-transcoding methods when cloud storage and cloud CDN is used (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 A.3: Incurred costs of different partial pre-transcoding methods when the variance of number of views for video streams increases (in US Dollar).

% 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

TABLE A.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.