Performance Analysis and Modeling of Video Transcoding Using Heterogeneous Cloud Services

Xiangbo Li, Mohsen Amini Salehi, *Member, IEEE,* Yamini Joshi,
Mahmoud K. Darwich, Brad Landreneau, and Magdy Bayoumi, *Fellow, IEEE*

Abstract—High-quality video streaming, either in form of Video-On-Demand (VOD) or live streaming, usually requires converting (*i.e.,* transcoding) video streams to match the characteristics of viewers' devices (*e.g.,* in terms of spatial resolution or supported formats). Considering the computational cost of the transcoding operation and the surge in video streaming demands, Streaming Service Providers (SSPs) are becoming reliant on cloud services to guarantee Quality of Service (QoS) of streaming for their viewers. Cloud providers offer heterogeneous computational services in form of different types of Virtual Machines (VMs) with diverse prices. Effective utilization of cloud services for video transcoding requires detailed performance analysis of different video transcoding operations on the heterogeneous cloud VMs. In this research, for the first time, we provide a thorough analysis of the performance of the video stream transcoding on heterogeneous cloud VMs. Providing such analysis is crucial for efficient prediction of transcoding time on heterogeneous VMs and for the functionality of any scheduling methods tailored for video transcoding. Based upon the findings of this analysis and by considering the cost difference of heterogeneous cloud VMs, in this research, we also provide a model to quantify the degree of suitability of each cloud VM type for various transcoding tasks. The provided model can supply resource (VM) provisioning methods with accurate performance and cost trade-offs to efficiently utilize cloud services for video streaming.

Index Terms—Heterogeneous Cloud service; Performance analysis; GOP Suitability Matrix; Video transcoding.

✦

1 INTRODUCTION

The way people watch videos has dramatically changed over the past years, from using traditional TV systems to streaming on desktops, laptops, and smartphones through the Internet. Based on the Global Internet Phenomena Report [1], 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 the whole Internet traffic by 2019 [2].

To have a high-quality video streaming experience, video contents, either in the form of Video On Demand (VOD) (*e.g.,* YouTube[1] or Netflix[2]) or live-streaming (*e.g.,* Livestream[3]), need to be converted based on the characteristics of the viewers' devices. That is, the original video has to be converted to a supported resolution, frame rate, video codec, and network bandwidth to match the viewers' display devices [3]. This conversion is termed *video transcoding* [4], which is a

computationally-heavy and time-consuming process.

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 [5], [6]. 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 [7]. The importance and prevalence of video streaming, in addition to its unique QoS demands, has motivated many researchers to investigate dedicated methods for resource allocation and provisioning of video streams (*e.g.,* [8], [9]).

Cloud providers offer abundant 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 [10], [11], video segmentation models [7], [12], scheduling [10], [13], and resource provisioning methods [7], [9]. However, these studies generally focus on 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.,*

- *Xiangbo Li is with Brightcove Inc. E-mail: xli@brightcove.com*
- *Mohsen Amini Salehi, Yamini Joshi, and Brad Landreneau are with the HPCC lab., School of Computing and Informatics, University of Louisiana at Lafayette, LA 70503, USA.*
  E-mail: {amini,yxj0845, bml6209}@louisiana.edu
- *Mahmoud K. Darwich is with the School of Engineering, Math and Technology, Navajo Technical University, NM 87313, USA.*
  E-mail: mdarwich@navajotech.edu
- *Magdy Bayoumi is with the Department of Electrical and Computer Engineering, University of Louisiana at Lafayette, LA 70503, USA.*
  E-mail: mab0778@louisiana.edu

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

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 *task-machine affinity* of a task type i on a machine (or VM) type j is defined as how tasks of type i matches (*i.e.,* can take advantage of) the architectural characteristics of machine type j. Higher affinity implies faster execution time of tasks type i on machine type j [14], [15]. For instance, particular transcoding tasks can be CPU-intensive 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.

Task scheduling and VM provisioning decisions are critical for SSPs to reduce cost while provide good service. 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 task-machine affinity of transcoding tasks with heterogeneous cloud VMs are required. Currently, there is no study of this kind available yet.

Expected Time to Compute (ETC) [16], [17] and *Estimated Computation Speed* (ECS) [18], [19] matrices are commonly used to model and explain the task-machine affinity. 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 arises is 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: **(1)** How can we recognize the task-machine affinity of different transcoding tasks with heterogeneous cloud VMs? **(2)** How to model the trade-off between performance and cost of heterogeneous VMs for different transcoding tasks?

To answer the first question, we need to find appropriate factors in video transcoding tasks that can determine the task-machine affinity of transcoding tasks with heterogeneous VMs. In particular, we investigate two factors, namely video transcoding operation and the video content type.

For that purpose, we analyze the task-machine 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. However, it is difficult to categorize video transcoding tasks based on their content type because the content type is not known prior to the execution of the tasks. Hence, in the next step, we find factors that indicate the video content type, 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 paper 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.

The rest of the paper is organized as follows. Section 2 provides background on video stream structure, video transcoding and the heterogeneous cloud VMs. In Section 3, we compare and analyze the task-machine affinity of transcoding tasks on heterogeneous cloud VMs. Suitability of transcoding tasks for heterogeneous VMs is proposed in Section 4. Section 5 reviews related works in the literature and position our work with respect to them. Finally, Section 6 summarizes findings of the paper.

2 BACKGROUND

2.1 Video Stream Structure

A Video stream, as shown in Figure 1, 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 related to the same scene in the video. A GOP starts with an `I` (intra) frame, followed by a number of `P` (predicted) frames or `B` (be-directional predicted) frames [20]. Each frame contains several *slices* that consist of a number of *macroblocks* which is the unit for video encoding and decoding operations. As each GOP can be processed independently, transcoding operation is commonly carried out at the GOP level [21]. Similarly, in this work, we assume that all the transcoding processes operate at the GOP level.

2.2 Video Transcoding

Video contents are initially captured with a particular format, spatial resolution, frame rate, and bit rate. SSPs usually have to adjust the original video based on the viewer'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* [3], [4]. Transcoding process includes decoding GOPS and re-encoding them in the new format. Accordingly, transcoding time is the sum of decoding and re-encoding times [22].

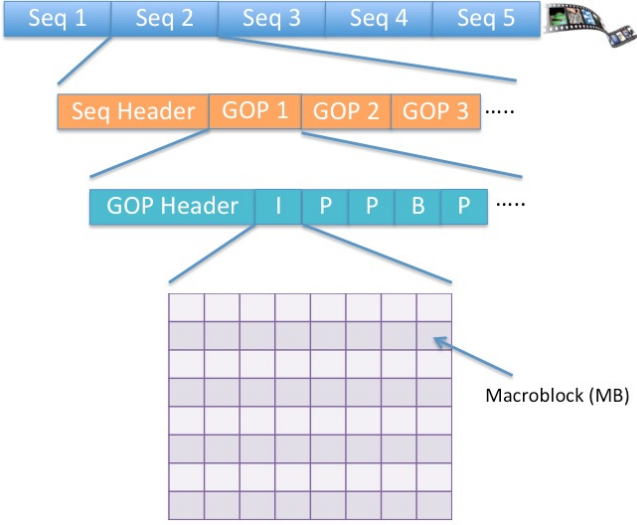Below, we provide more details on the nature of processing in different transcoding operations:

Fig. 1: The structure of a video stream. Each sequence includes multiple GOPs. Frames of a GOP are of I (intra), P (predicted), or B (be-directional predicted) types.

2.2.1 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, SSPs usually need to change the bit rate of video streams to ensure smooth streaming [23]. Dynamic bit rate adjustment of video streams is also known as Adaptive video streaming [24].

2.2.2 Spatial Resolution Reduction

The spatial resolution indicates the dimensional size of a video. The dimensional size of an original video stream does not necessarily match to 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 [25].

2.2.3 Temporal Resolution Reduction

Temporal resolution reduction happens when the viewer's device only supports a lower frame rate. In this situation, the SSP has to drop some frames. Due to the dependency between frames, dropping frames may cause motion vectors become invalid for the incoming frames. Details of methods for temporal resolution reduction can be found in [26].

2.2.4 Video Compression Standard Conversion

There is a wide variety of video compression standards (codec) for video files —from MPEG2 [27], to H.264 [20], and to the most recent one, HEVC [28]. 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. Viewer's 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 [29].

2.3 Video Content Type

Each GOP covers one scene in a video and utilizes still background content in the video to reduce its size. Accordingly, based on the frequency of scene changes, video contents can be categorized into three types: *slow motion*, *fast motion*, and *mixed motion*.

In *slow motion* videos, the scene changes slowly and the background remains still. Therefore, GOPs of such videos include many frames and are large in size. In contrast, the scene changes of *fast motion* videos (*e.g.,* action movies) are dramatic. These videos contain many GOPs, however, each GOP includes few frames, hence, it is small in size. A *mixed motion* video includes a combination of both fast and slow motion scenes, thus, includes GOPs with a variety of sizes.

2.4 Heterogeneous VMs 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 cluster of heterogeneous VMs to process high performance computations in the cloud. Heterogeneous systems are categorized as *consistent* and *inconsistent* [18] environments. The former refers to environments in which some machines (VMs) are 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 [30]. 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 inconsistently heterogeneous environments.

In the case of Amazon EC2 cloud, 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 are the least expensive one and have lower computing power in comparison with other VM types. Generally, to process a large set of tasks, either many or few of these VMs should be allocated for a long time [31].
- **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 for high-traffic web application servers, batch processing, video encoding, and

high performance computing applications (*e.g.,* genome analysis and high-energy physics) [32].

- **Memory Optimized VMs:** Memory-Optimized VM type is designed for processing tasks with large memory demand. This 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 [33] 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). Many large-scale simulations, such as computational chemistry, rendering, and financial analysis are conducted on GPU-Optimized VMs [34].

- **Storage Optimized and Dense Storage VMs:** These VM types 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 [35].

We perform this research by using VM types offered by the Amazon cloud provider. The reason we chose Amazon is that it is the mainstream cloud provider and many video SSPs utilize its services [37]. However, we would like to note that the analysis provided in this work is general and can be applied to any heterogeneous computing (HC) environment.

3 PERFORMANCE ANALYSIS OF TRANSCODING OPERATIONS ON HETEROGENEOUS CLOUD VMS

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 2.4).

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 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[4].

To analyze the transcoding time, we utilized a set of benchmark videos. The benchmarking videos are publicly available for reproducibility purposes[5]. Videos in the benchmark are diverse both in terms of the content types and length. The benchmark includes a combination of slow, fast, and mixed

4. https://aws.amazon.com/ec2/instance-types/
5. The videos can be downloaded from: https://goo.gl/TE5iJ5

TABLE 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

motion video content types. The length of the videos in the benchmark varies in the range of [10, 600] Seconds. The size and frame number of the benchmark videos ranges from 5MB to 313MB, and 240 to 10464, respectively.

We used `FFmpeg`[6], which is an open source utility, to transcode the videos. State-of-the-art FFmpeg transcoder is a cascaded transcoder with sequential transcoding algorithm, that means the incoming source video stream is fully decoded before re-encoding into the target video stream with the desired codec, bitrate, and frame rate [3], [4]. For each one of the benchmarking videos, four different transcoding operations, namely codec 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)[7]. The mean transcoding time on each VM for a given GOP is considered for comparison and analysis of this paper.

3.2 Analyzing the Execution Time of Different Video Transcoding Operations

The first question we need to answer is to identify if a certain transcoding operation has a stronger task-machine affinity with a particular cloud VM type.

To answer this question, we compared the transcoding time (execution time) of various transcoding operations using different VM types. 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. We should note that, because GOPs are transcoded independently and there are diverse types of video contents in the repository, the nine GOPs are representative of other GOPs in the benchmark.

Figure 2 shows the transcoding time of different transcoding operations on 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 2a, 2b, 2c, and 2d 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

6. https://ffmpeg.org
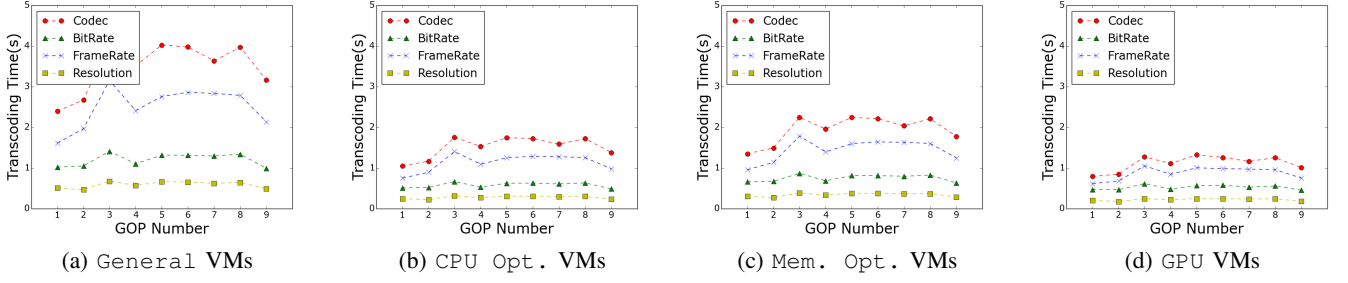7. The workload traces are available at: https://goo.gl/B6T5aj

Fig. 2: 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.

implies decoding the original format of the video and then, encoding it to a new 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 [38], [39] 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.

3.3 Analyzing the Task-Machine Affinity of Video Transcoding Operations with Heterogeneous VMs
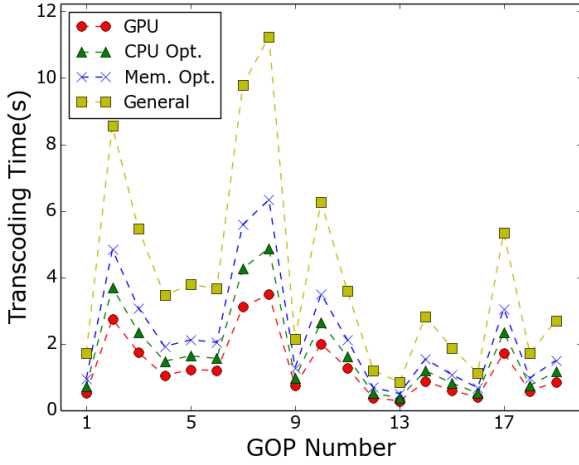

Fig. 3: Execution time of codec transcoding for a video in the benchmark on different VM types.

As mentioned in Section 2.4, 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 task-machine affinity of video transcoding operations with heterogeneous cloud VMs?

As we noticed in Section 3.2, although execution times of various video transcoding operations are different, codec transcoding has the highest execution time and changing spatial resolution generally has the lowest execution time. Considering this pattern, to study the task-machine affinity of

video transcoding on heterogeneous VMs, we only 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 expresses the analysis for one video[8] 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 1).

More importantly, in Figure 3, we observe that the transcoding times of different GOPs 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 normalized the transcoding time of GOP i on a given VM type, by dividing it by the transcoding time of GOP i on `GPU`. The result of this analysis is shown in Figure 4. In all sub-figures of Figure 4, the horizontal axis shows the performance ratio and the vertical access shows the frequency of that ratio across all GOPs in a video. That is, the number of times each performance ratio has occurred for all GOPs. We fit a Bell curve on the histograms of these sub-figures and the results conform with the Normal distribution. Mean and Standard Deviation of the fitted Normal distribution, are as follows:

1) In Sub-figure 4a, performance ratio of `General` VM lies within the range 2.781 ± 1.524.

2) In Sub-figure 4b, performance ratio of `CPU Opt.` VM lies within the range 1.263 ± 0.508.

8. This is `big_buck_bunny_720p_h264_02tolibx264` video in the benchmark.
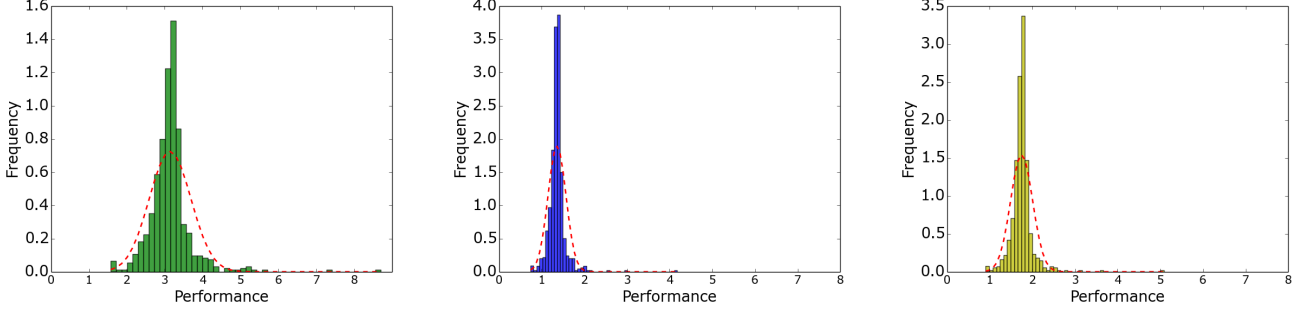
(a) Performance ratio of `General` to `GPU`. (b) Performance ratio of `CPU-Opt.` to `GPU`. (c) Performance ratio of `Mem-Opt.` to `GPU`.

Fig. 4: 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.

3) In Sub-figure 4c, performance ratio of `Mem. Opt.` VM lies within the range 1.608 ± 0.652.

TABLE 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: 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%

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 2. We see that 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, the percentage of tasks that have performance ratio lower than 1.2 are reported.

Summary of our observations in this part:

1) 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). We note that 0.24 second is relatively low and negligible when compared with the delay caused by network.

2) 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.

3) From the two previous observations, we conclude that, in a cloud environment with heterogeneous VMs, making use of expensive VM types for tasks with short execution time is not beneficiary. However, understanding the exact execution time threshold requires benchmarking in that particular context and study the performance cost ratio of using different VM types.

4) According to Figure 4, 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 for video transcoding.

3.4 Analyzing the Impact of Video Content Type on Transcoding Time

As we observed in the previous section, the transcoding time of a GOP can vary significantly on different VM types. For instance, in Figure 3, transcoding time difference between `GPU` and `CPU Opt.` VM types for GOP 8 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 5.

Figure 5a shows that the transcoding times of the slow motion videos are 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 5b shows that the outperformance of `GPU` VM is not statistically and practically significant when

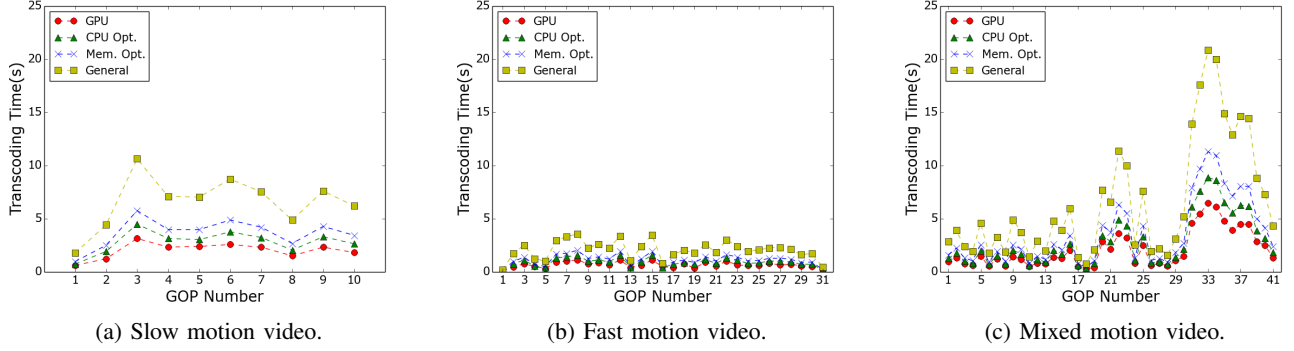(a) Slow motion video.　　　　(b) Fast motion video.　　　　(c) Mixed motion video.

Fig. 5: 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.

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 5c. 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 in size. 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 numbers of GOPs that are large in size (*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.

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

In Section 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. In this section, we investigate further the factors that influence GOP transcoding time on different VM types.

As we noticed in Section 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 the 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 6 and 7, respectively. The vertical axes show the transcoding times of GOPs (in seconds).
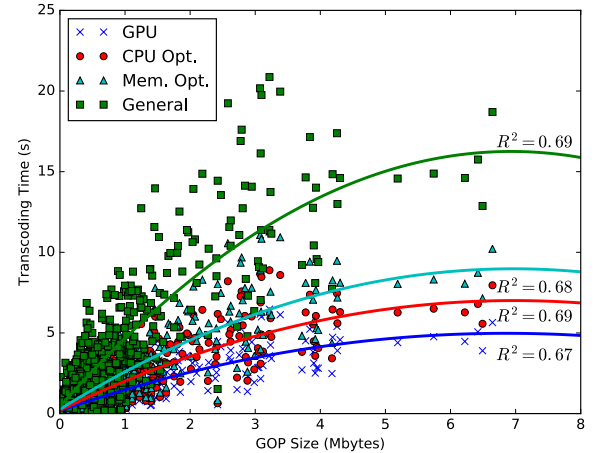

Fig. 6: Second degree regression to study the influence of GOP size on the transcoding time.

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. As we can, 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 a 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, the number of frames provides a
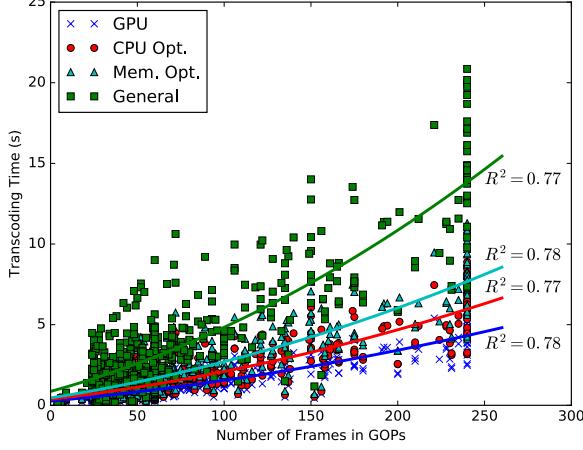
Fig. 7: Second degree regression to study the influence of number of frames in a GOP on the transcoding time.

stronger regression with transcoding time.

In Figure 7, 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 numbers 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.

4 PERFORMANCE COST TRADE-OFF OF TRANSCODING ON HETEROGENEOUS VMS

4.1 Overview

VM types offered by cloud providers are heterogeneous both in terms of performance and cost [40]. Hence, allocating VMs that are cost- and performance-efficient for transcoding tasks is challenging.

As we discussed in Section 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.

4.2 Modeling Performance Cost Trade-Off of Transcoding Tasks on Heterogeneous VMs

Recall from Table 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 the incurred 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 [41]. As shown in Figure 8, 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.
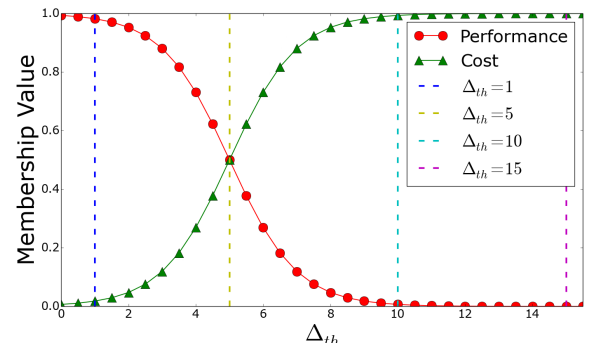
Fig. 8: Membership functions for performance and cost preferences. The user-provided values for the cost or performance are considered as the membership value (vertical axis). Then, the corresponding value on the horizontal axis is considered as Δ_{th}

Value of the user's performance (or cost) preference is considered as the membership value of the fuzzy membership function (vertical axis in Figure 8) and is used to obtain the performance threshold gap (horizontal axis in Figure 8). More

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**
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

(a) Suitability Matrix, when p = 98%, c = 2%, and Δ_{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
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

(b) Suitability Matrix, when p = 50%, c = 50%, and Δ_{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
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

(c) Suitability Matrix, when p = 1%, c = 99%, and Δ_{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**
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

(d) Suitability Matrix, when p = 0.01%, c = 99.99%, and Δ_{th} = 15

TABLE 4: 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 4a to `General` type (cost-oriented VM) in Table 4d.

specifically, by using the performance preference (p), we can obtain Δ_{th} based on Equation (1)[9].

$$\Delta_{th} = \frac{\ln \frac{1-p}{p}}{\alpha} + \beta \qquad (1)$$

where α is the inflection point in the membership function and β is the slope at α. In Figure 8, we experimentally obtained the values of α and β equal to 1 and 5, respectively.

Based on the value of Δ_{th}, we can determine the trade-off between performance 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 2 that encompasses both the performance and cost factors.

---

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

The first part, in Equation 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 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.

$$W_i = \frac{\Delta_{th} - \Delta_i}{\sum_{n=1}^{N} \Delta_n} \cdot \left(1 - \frac{\varphi_i}{\sum_{n=1}^{N} \varphi_n}\right) \qquad (2)$$

(a) Comparison of startup delay (b) Comparison of deadline miss rate (c) Comparison of cost
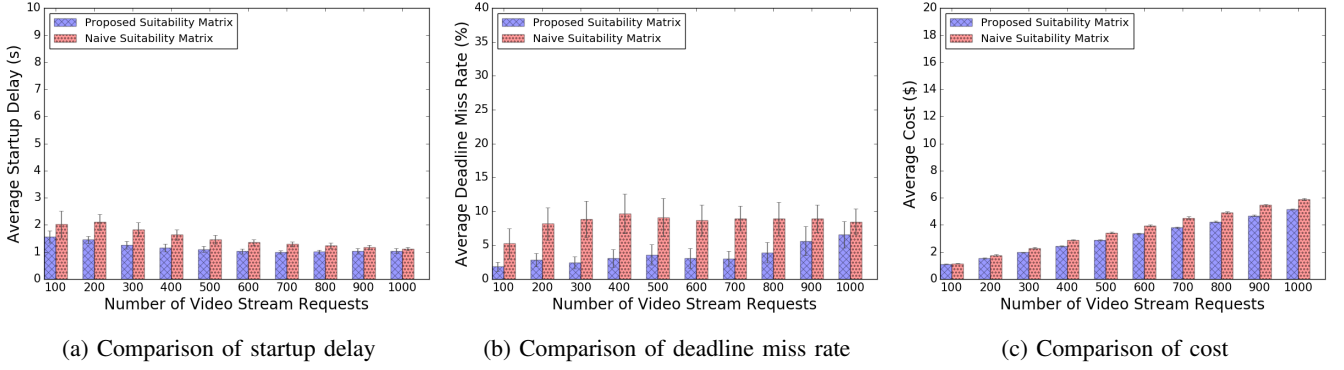
Fig. 9: Performance and cost comparison when our proposed suitability matrix is used against the a naïve suitability matrix. Horizontal axes in all subfigures show the number of streaming tasks.
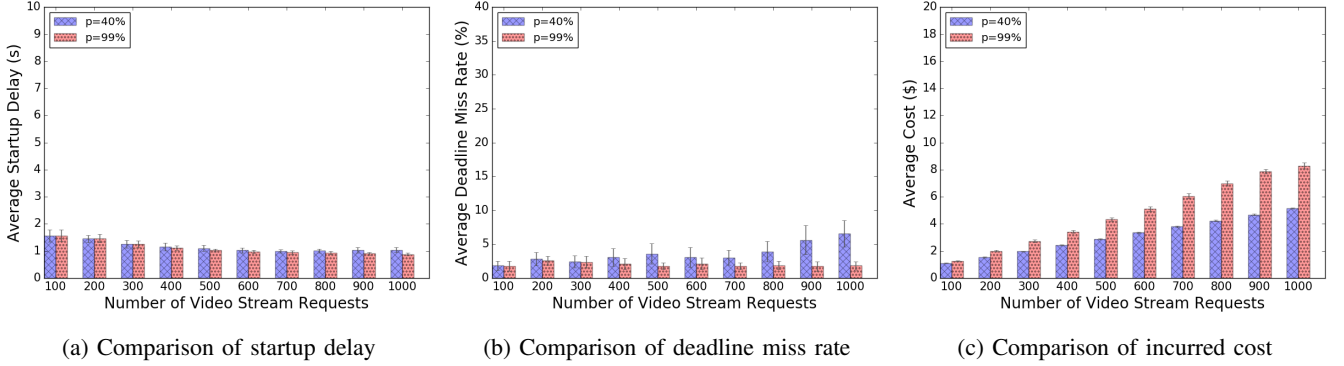

(a) Comparison of startup delay (b) Comparison of deadline miss rate (c) Comparison of incurred cost

Fig. 10: Performance and cost comparison of our proposed suitability matrix with different performance rates, p=40% versus p=99%. Horizontal axes in all subfigures show the number of streaming tasks.

To normalize the value of W_i and determine the final suitability values, denoted S_i, between [0, 1] we use Equation (3) as follows:

$$S_i = \frac{W_i - \max_i(W_i)}{\max_i(W_i) - \min_i(W_i)} \qquad (3)$$

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.

4.3 Case Study of the Trade-Off Model

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 4a 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 4b, 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 4c and 4d). 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.

4.4 Performance Evaluation

In the experiments of this section, we used CloudSim [42], a discrete event simulator, to model our system and evaluate performance of the scheduling methods and VM provisioning policies. We modeled the system based on the characteristics and cost of VM types in Amazon EC2. We measured the startup delay, deadline miss rate of video streams, and the incurred cost of using cloud VMs to process different number of streaming tasks (from 100 or 1000) arriving during the same time period[10]. For the sake of accuracy, each experiment has been conducted 30 times and the mean and 95% confidence interval of the results are reported. For this experiment, we consider the performance ratio p=40% (and cost ratio c=60%).

To demonstrate the efficacy of our proposed trade-off model, in the first experiment, we compare the performance and the

---

10. Details of the generated workload can be downloaded from https://goo.gl/TE5iJ5

incurred cost when the scheduling method uses the proposed suitability matrix against a naïve suitability matrix that has been proposed in [8].

The naïve method operates simply based on a trade-off between the performance (T_i) and the cost (C_i) for a given GOP on VM type i, as shown in Equation (4), while it does not consider the performance tolerance that user can decide like our proposed approach.

$$S_i = k \cdot T_i + (1 - k) \cdot C_i \qquad (4)$$

As we can see in Figure 9, the resource allocation system that uses our proposed suitability matrix leads to a lower startup delay and a lower deadline miss rate at even a lower cost. The reason is our proposed method can more accurately assign GOP types based on user's preference.

To further investigate the impact of SSP's preference on the performance (and cost) when our proposed suitability matrix is deployed, in the second experiment, we compared the performance and the incurred cost with two performance ratios, namely $p=40\%$ and $p=99\%$. Figure 10 expresses that for the higher value of performance ratio, both the startup delay and deadline miss rate is improved. The improvement is more remarkable when there are more tasks in the system. In addition, we can see that the incurred cost also significantly increases for a higher performance ratio. The experiment testifies that the performance and incurred cost resulted from deploying the proposed suitability matrix conforms with the discretion of the streaming service provider.

5 RELATED WORK

Several studies explored the performance analysis of heterogeneous cloud services [31], [32], [43]. Iosup *et al.* [43] and Jackson *et al.* [32] studied application-oriented performance analysis using heterogeneous cloud services. The results show that although cloud services have their own drawbacks in terms of communication and processing delays, utilizing cloud services is a viable solution for processing workloads that need resources instantly and temporarily. Lee *et al.* [31] investigate 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 analyzing video transcoding tasks on heterogeneous VM types in clouds.

Expected Time to Compute (ETC) [16], [17], [30] and *Estimated Computation Speed* (ECS) [18], [19] matrices are commonly used to explain the affinity of different tasks types on heterogeneous machines. These matrices are utilized for more efficient task scheduling and VM allocation. However, the definition of both ETC and ECS only considers execution time as the performance metric and ignores the cost heterogeneity across different VM types. Our proposed *Suitability Matrix* extends the idea of ETC matrices by including both performance and cost metrics.

Video transcoding is a computationally expensive and time-consuming operation. Techniques, architectures, and the challenges of video transcoding were investigated by Ahmad *et al.* [3] and Vetro *et al.* [4]. With the rise of cloud computing,

Streaming Service Providers (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 11. Challenges of cloud-based transcoding for VOD was studied in [9], [13]. Studies have been concentrated on both pre-transcoding [9], [12], [13], [44], [45], on-demand transcoding [7], [46], [47] and live streaming [11], [48], [49].
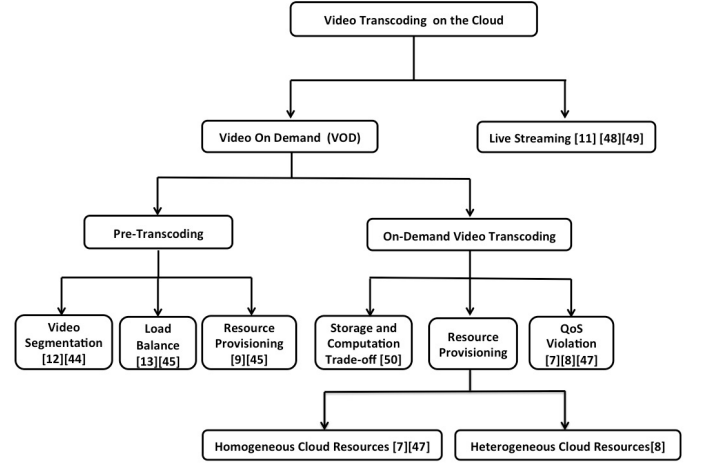

Fig. 11: A taxonomy of researches undertaken on video transcoding using cloud services.

For pre-transcoding, the research focus is mainly on video segmentation [12], [44], load balance [13], [45], and resource provisioning [9], [45], while the 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 [7], [8], [47]. Li *et al.* [7], [8] propose the CVSS architecture to efficiently transcode video in an on-demand manner on homogeneous and heterogeneous cloud VMs, respectively. With proper scheduling and resource provisioning policy, CVSS provides low startup delay and playback jitter. Li *et al.* [47] 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.

Jokhio *et al.* [50] 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.* [51] 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. Kathpal *et al.* [52] developed cost metrics that enable comparing storage versus

compute costs and determine when an on-demand transcoding can be cost-effective. They also analyze how such a solution can be deployed in a storage system based on the access pattern information or online algorithms when such access patterns are not available.

The idea of cloud-based video transcoding has also has been applied to live video streaming [11], [48]. Timmerer*et al.* [53] 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 [49] standard. Lai *et al.* [54] design a cloud-assisted real-time transcoding mechanism based on the HLS protocol [55], they implement the bandwidth recoder, segment transrater, and segment redirector on the server. They provide an instant analysis of the online quality between client and server without changing the HLS server architecture and the optimum media quality.

With the trend of video transcoding using cloud services, a better understanding the performance of different video transcoding operation on heterogeneous VMs is necessary. Transcoding time estimation plays an important role in both efficient scheduling and resource provisioning. Deneke *et al.* [56] utilize machine learning methods based on the video characteristics (*e.g.,* resolution, frame rate, and bit rate) to predict the transcoding time. Seo *et al.* [22] focus 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 [22], [56] do not consider the diversity of heterogeneous environment of cloud services. Our work provides a deep performance analysis and transcoding time estimation for different transcoding operations on heterogeneous VM types, which is beneficial for cost and performance efficient video transcoding scheduling and resource provisioning using cloud services.

6 SUMMARY AND DISCUSSION

With the emergence of on-demand video transcoding on the cloud, it is crucial to study the video transcoding tasks and influential factors on their execution times. In addition, it is necessary to come up with a trade-off between performance and cost of using cloud services. The trade-off becomes further complicated when we consider the heterogeneity of computational services (VMs) offered by cloud providers. To understand the affinity of different transcoding tasks and heterogeneous VM types we provided a detailed study and analysis of different transcoding operations on heterogeneous VMs. In summary, the main findings of our research are as follows:

1) The execution times of different transcoding operations follow a pattern: video codec and adjusting frame rate transcoding require more computation time than bit rate and spatial resolution transcoding.

2) Although `GPU` VM type mostly provides a faster execution time than other VM types, in some cases the execution time difference is negligible. In particular,

we observed that when transcoding tasks are categorized based on transcoding type, up to 63% of bit rate transcoding tasks can be executed on VM types other than `GPU` with nearly the same transcoding time (see Table 3) while incurring a significantly lower cost.

3) We learned that the execution time of the transcoding operation on heterogeneous VMs has a correlation with the video content type. GOPs that contain slow motion video content are larger in size and include more frames in compare to GOPs of fast motion videos. Thus, GOPs with slow motion video content can benefit from computationally powerful VMs whereas fast motion ones can be executed on less powerful and more cost efficient VMs with a similar performance.

4) Cloud VMs exhibit inconsistent heterogeneity behavior in executing video transcoding tasks. However, the inconsistent behavior is more related to video content type rather than the type of transcoding operation. As such, video transcoding tasks (GOPs) are suggested to be categorized based on their content type to gain more from heterogeneous VMs offered by cloud providers.

5) As identifying GOPs' content types prior to execution is difficult, we can use the number of frames (or frame size) in the GOP as an intuitive factor that indicates the content type of transcoding tasks.

6) By considering both the performance and cost heterogeneity of different VM types, we provided a model that, identifies the degree of suitability of each VM type for a given GOP. The provided model operates based on the SSP performance and cost preference. Suitability matrices can supply resource allocation and scheduling methods with accurate performance and cost trade-offs to utilize appropriate VMs for video transcoding.

7) Evaluations show that in comparison to naïve method in [8], our suitability matrix provides a lower startup delay and a lower deadline miss rate at a lower cost.

REFERENCES

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

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

[3] 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.

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

[5] 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*, pp. 39–44, Apr. 2017.

[6] E. Baik, A. Pande, Z. Zheng, and P. Mohapatra, "VSync: Cloud based video streaming service for mobile devices," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications*, ser. INFOCOM '16, pp. 1–9, Apr. 2016.

[7] 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, pp. 106–115, May 2016.

[8] X. Li, M. A. Salehi, M. Bayoumi, N. F. Tzeng, and R. Buyya, "Cost-efficient and robust on-demand video transcoding using heterogeneous cloud services," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 29, no. 3, pp. 556–571, March 2018.

[9] 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*, ser. PDP '13, pp. 254–261, Feb. 2013.

[10] T. Deneke, H. Haile, S. Lafond, and J. Lilius, "Video transcoding time prediction for proactive load balancing," in *IEEE International Conference on Multimedia and Expo*, ser. ICME '14, pp. 1–6, July 2014.

[11] X. Li, M. A. Salehi, and M. Bayoumi, "VLSC: Video Live Streaming Using Cloud Services," in *Proceedings of the 6th IEEE International Conference on Big Data and Cloud Computing Conference*, ser. BD-Cloud '16, pp. 595–600, Oct. 2016.

[12] F. Jokhio, T. Deneke, S. Lafond, and J. Lilius, "Analysis of video segmentation for spatial resolution reduction video transcoding," in *Proceedings of the 19th IEEE International Symposium on Intelligent Signal Processing and Communications Systems*, ser. ISPACS '11, pp. 1–6, Dec. 2011.

[13] 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*, ser. ISCAS '13, pp. 2864–2867, May 2013.

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

[15] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *Journal of parallel and distributed computing*, vol. 59, no. 2, pp. 107–131, 1999.

[16] 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*, ser. PDPTA '12, pp. 1, June 2012.

[17] 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, Sep. 2000.

[18] 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 25th IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, ser. IPDPSW '11, pp. 34–44, May 2011.

[19] 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*, ser. IPDPSW '10, pp. 1–9, Apr. 2010.

[20] 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.

[21] 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*, ser. ISCAS '12, pp. 2905–2908, May 2012.

[22] D. Seo, J. Kim, and I. Jung, "Load distribution algorithm based on transcoding time estimation for distributed transcoding servers," in *Proceedings of International Conference on Information Science and Applications*, ser. ICISA '10, pp. 1–8, Apr. 2010.

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

[24] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proceedings of the 8th International Conference on emerging Networking Experiments and Technologies*, ser. CoNEXT '12, pp. 97–108, June 2012.

[25] N. Bjork and C. Christopoulos, "Transcoder architectures for video

coding," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 1, pp. 88–98, Feb. 1998.

[26] 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.

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

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

[29] 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*, ser. ISM '09, pp. 385–391, Dec. 2009.

[30] 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," *Journal of Parallel and Distributed Computing (JPDC)*, vol. 97, pp. 96–111, June 2016.

[31] 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, pp. 4, Oct. 2011.

[32] 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*, ser. CloudCom '10, pp. 159–168, Nov. 2010.

[33] 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.

[34] 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.

[35] 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.

[36] 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*, ser. ICPP '11, pp. 315–324, Sep. 2011.

[37] 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 the 31st Annual IEEE International Conference on Computer Communications*, ser. INFOCOM '12, pp. 1620–1628, Mar. 2012.

[38] 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.

[39] P. Yin, M. Wu, and B. Liu, "Video transcoding by reducing spatial resolution," in *Proceedings of International Conference on Image Processing*, ser. ICIP '00, vol. 1, pp. 972–975, Sep. 2000.

[40] 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*, ser. AINA '10, pp. 27–33, Apr. 2010.

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

[42] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, pp. 23–50, Aug. 2011.

[43] 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 (TPDS)*, vol. 22, no. 6, pp. 931–945, June 2011.

[44] 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.

[45] 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*, ser. CCGrid '13, pp. 482–489, May 2013.

[46] 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, pp. 600–603, May 2016.

[47] 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*, ser. NOSSDAV '12, pp. 33–38, June 2012.

[48] 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.

[49] 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.

[50] 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*, ser. SEAA '13, pp. 365–372, Sep. 2013.

[51] 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 20th IEEE Symposium on Computers and Communication*, ser. ISCC, '15, pp. 943–948, July 2015.

[52] A. Kathpal, M. Kulkarni, and A. Bakre, "Analyzing compute vs. storage tradeoff for video-aware storage efficiency," in *Proceedings of the 4th USENIX Workshop on Hot Topics in Storage and File Systems*, June 2012, pp. 13–18.

[53] C. Timmerer, D. Weinberger, M. Smole, R. Grandl, C. Müller, and S. Lederer, "Live transcoding and streaming-as-a-service with MPEG-DASH," in *IEEE International Conference on Multimedia and Expo Workshops*, ser. ICMEW '15, pp. 1–4, June 2015.

[54] 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.

[55] T. Stockhammer, "Dynamic adaptive streaming over http: standards and design principles," in *Proceedings of the 2nd annual ACM conference on Multimedia systems*, ser. MMSYS '11, pp. 133–144, Feb. 2011.

[56] T. Deneke, H. Haile, S. Lafond, and J. Lilius, "Video transcoding time prediction for proactive load balancing," in *2014 IEEE International Conference on Multimedia and Expo*, ser. ICME '14, pp. 1–6, July 2014.

Yamini Joshi received the B.S. in Computer Science from Rajasthan Technical University, Kota, India. She is currently a Master students in the School of Computing and Informatics at University of Louisiana Lafayette. Her research focus is on Distributed and Cloud Computing and Big Data Analytics.

Mahmoud K. Darwich received the B.S. degree in electrical engineering, communications and electronics, from Beirut Arab University, Lebanon in 2006, the M.S. degree and the Ph.D. degree, in computer engineering from University of Louisiana at Lafayette in 2013 and 2017 respectively. He is currently a faculty member at Navajo Technical University. His research interests include Cloud Computing, Video transcoding on Cloud, and VLSI circuits.

Brad Landreneau is currently a Bachelor student and an active member of the High Performance Cloud Computing (HPCC) laboratory, School of Computing and Informatics at University of Louisiana Lafayette, USA. His research interests are Distributed and Cloud Computing, and Android Application Development.

Xiangbo Li received his Ph.D. degree in computer engineering from University of Louisiana at Lafayette in 2016. He is currently working as video engineer at Brightcove Inc., a cloud based online video platform company. His areas of expertise include cloud-based video encoding, transcoding, packaging and server side ad insertion.

Magdy Bayoumi received the BSc and MSc degrees in electrical engineering from Cairo University, Egypt, the MSc degree in computer engineering from Washington University, St. Louis, and the Ph.D. degree in electrical engineering from the University of Windsor, Ontario. He was the Vice President for Conferences of the IEEE Circuits and Systems (CAS) Society. He is the recipient of the 2009 IEEE Circuits and Systems Meritorious Service Award and the IEEE Circuits and Systems Society 2003 Education Award.

Mohsen Amini Salehi received his Ph.D. in Computing and Information Systems from Melbourne University, Australia, in 2012. He is currently an Assistant Professor and director of the High Performance Cloud Computing Laboratory, School of Computing and Informatics at University of Louisiana Lafayette, USA. His research focus is on different aspects of Distributed and Cloud computing including heterogeneity, load balancing, virtualization, resource allocation, energy-efficiency, and security.