A Computation and Storage Trade-off Strategy for Cost-Efficient Video Transcoding in the Cloud

Fareed Jokhio*†, Adnan Ashraf*‡§, Sébastien Lafond*‡, Johan Lilius*‡
* Department of Information Technologies, Åbo Akademi University, Turku, Finland.
Email: {fjokhio, aashraf, slafond, jolilius}@abo.fi
† Quaid-e-Awam University of Engineering, Science & Technology, Nawabshah, Pakistan.
‡ Turku Centre for Computer Science (TUCS), Turku, Finland.
§ Department of Software Engineering, International Islamic University, Islamabad, Pakistan.

***Abstract*—Video transcoding refers to the process of converting a compressed digital video from one format to another. Since it is a compute-intensive operation, transcoding of a large number of on-demand videos requires a large scale cluster of transcoding servers. Moreover, storage of multiple transcoded versions of each source video requires a large amount of disk space. Infrastructure as a Service (IaaS) clouds provide virtual machines (VMs) for creating a dynamically scalable cluster of servers. Likewise, a cloud storage service may be used to store a large number of transcoded videos. Moreover, it may be possible to reduce the total IaaS cost by trading storage for computation, or vice versa. In this paper, we present a computation and storage trade-off strategy for cost-efficient video transcoding in the cloud called cost and popularity score based strategy. The proposed strategy estimates computation cost, storage cost, and video popularity of individual transcoded videos and then uses 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. It is demonstrated in a discrete-event simulation and is evaluated in a series of experiments involving semisynthetic and realistic load patterns.**

***Keywords*-Video transcoding; computation and storage trade-off; cost-efficiency; cloud computing**

I. Introduction

With an ever increasing number of digital videos delivered everyday via the Internet, the number of video formats and video codecs used for digital video representation are also increasing rapidly. Moreover, since video streaming of a large number of videos requires a lot of server-side resources, digital videos are often stored and transmitted in compressed formats to conserve storage space and communication bandwidth. With the emergence of a large number of video compression techniques and packaging formats, such as MPEG-4 [1] and H.264 [2], the diversity of digital video content representation has grown even faster. However, a client-side device may support only a small subset of the existing video formats. Therefore, an unsupported format needs to be converted into one of the supported formats before the video could be played.

Video transcoding refers to the process of converting a compressed digital video from one format to another. In addition to format conversion, video contents are sometimes also altered in terms of bit-rate and resolution to meet the network bandwidth requirements or capabilities of the client-side device. Modern client-side devices also use a large variety of display sizes and resolutions, which has necessitated the need to simultaneously transcode video streams to different video formats, spatial resolutions, and bit-rates. Thus, video transcoding [3], [4], [5], [6], [7], [8], [9], [10] is used for bit-rate reduction, spatial resolution reduction, temporal resolution reduction, and video-coding format conversion.

Since video transcoding is a compute-intensive operation, transcoding of a large number of on-demand videos requires a large scale cluster of transcoding servers. Moreover, storage of multiple transcoded versions of each source video requires a large amount of disk space. Infrastructure as a Service (IaaS) clouds provide virtual machines (VMs) for creating a dynamically scalable cluster of servers. Likewise, a cloud storage service may be used to store a large number of transcoded videos. Determining the number of VMs and the amount of storage to provision from an IaaS cloud is an important problem. The exact number of VMs and the exact amount of storage needed at a specific time depend upon the incoming load from service users and their performance requirements. In our previous work [11], we proposed a prediction-based dynamic resource allocation approach for video transcoding in cloud computing. However, finding a cost-efficient computation and storage trade-off strategy for video transcoding in cloud computing is still an open problem. The objective is to reduce the total IaaS cost by trading storage for computation, or vice versa. In this paper, we investigate the computation and storage cost trade-off for video transcoding in the cloud and present a cost-efficient strategy called cost and popularity score based strategy. The proposed strategy estimates computation cost, storage cost, and video popularity of individual transcoded videos and then uses this information to make decisions on how long a video should be stored or how frequently it should be re-transcoded from its source video.

We proceed as follows. Section II presents the system architecture of an on-demand video transcoding service and sets the context for the proposed computation and storage trade-off strategy. The proposed strategy is presented in Section III. Section IV describes experimental design and setup. The results of the experimental evaluation are presented in

Section V. In Section VI, we discuss important related works before concluding in Section VII.

II. System Architecture

The system architecture of the cloud-based on-demand video transcoding service is shown in Figure 1. It consists of a *streaming server*, a *video splitter*, a *video merger*, a *video repository*, a dynamically scalable cluster of *transcoding servers*, a *load balancer*, a *master controller*, and a *load predictor*. The video requests and responses are routed through the *streaming server*. Since the main focus of this paper is on computation and storage trade-off for video transcoding, we assume that the *streaming server* is not a bottleneck.

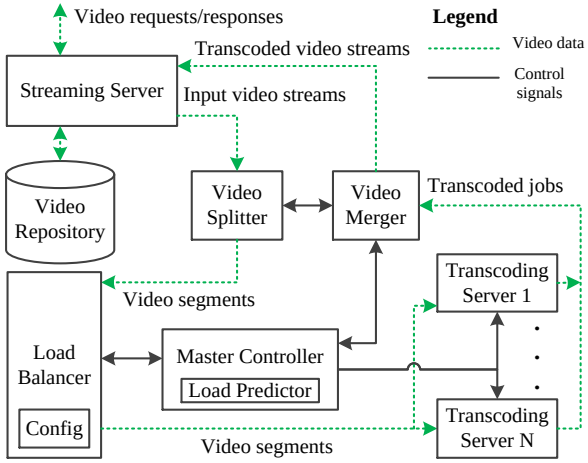

Fig. 1. System architecture

The video streams in certain compressed formats are stored in the *video repository*. The *streaming server* accepts video requests from users and checks if the required video is available in the *video repository*. If it finds the video in the desired format and resolution, it starts streaming the video. However, if it finds that the requested video is stored only in another format or resolution than the one desired by the user, it sends the video for segmentation and subsequent transcoding. Then, as soon as it receives the transcoded video from the *video merger*, it starts streaming the video.

After each transcoding operation, the computation and storage trade-off strategy determines if the transcoded video should be stored in the *video repository* or not. Moreover, if a transcoded video is stored, then the trade-off strategy also determines the duration for which the video should be stored. Therefore, it allows us to trade computation for storage or vice versa in order to reduce the total operational cost and to improve performance of the transcoding service.

The *video splitter* splits the video streams into smaller segments called jobs, which are placed into the job queue. A compressed video consists of three different types of frames namely, *I*-frames (intracoded frames), *P*-frames (predicted frames), and *B*-frames (bi-directional predicted frames). Due to inter-dependencies among different types of frames, the video splitting or segmentation is performed at the key frames, which are always *I*-frames. An *I*-frame followed by *P* and *B* frames is termed as a group of pictures (GOP). GOPs represent atomic units that can be transcoded independently of one another [11]. Video segmentation at GOP level is discussed in more detail in [12] and [13].

The *load balancer* distributes load on the transcoding servers. In other words, it routes and load balances transcoding jobs on the *transcoding servers*. It maintains a configuration file, which contains information about *transcoding servers* that perform the transcoding operations. As a result of dynamic resource allocation and deallocation operations, the configuration file is often updated with new information. The *load balancer* serves the jobs in FIFO (First In, First Out) order. It implements one or more job scheduling policies, such as, the *shortest queue length* policy, which selects a *transcoding server* with the shortest queue length and the *shortest queue waiting time* policy, which selects a *transcoding server* with the least queue waiting time.

The actual transcoding is performed by the transcoding servers. They get compressed video segments, perform the required transcoding operations, and return the transcoded video segments for merging. A *transcoding server* runs on a dynamically provisioned VM. Each *transcoding server* processes one or more simultaneous jobs. When a transcoding job arrives at a *transcoding server*, it is placed in the server's queue from where it is subsequently processed.

The *master controller* acts as the main controller and resource allocator. It implements prediction-based dynamic resource allocation and deallocation algorithms [11] and one or more computation and storage tarde-off strategies. The resource allocation and deallocation is mainly based on the target play rate of the video streams and the predicted transcoding rate of the transcoding servers. For load prediction, the *master controller* uses *load predictor* [14]. The *video merger* merges the transcoded jobs into video streams, which form video responses. Our resource allocation and load prediction algorithms are described in detail in [11] and [14]. In this paper, our primary focus is on a cost-efficient computation and storage trade-off strategy.

III. Proposed Strategy

In this section, we present the proposed computation and storage trade-off strategy. For the sake of clarity, we provide a summary of the notations in Table I.

The proposed cost and popularity score based strategy estimates the computation cost, the storage cost, and the video popularity of individual transcoded videos and then uses 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. In an on-demand video streaming service, the source videos are usually high quality videos that comprise the primary datasets. Therefore, irrespective of their computation and storage costs, they are never deleted from the *video repository*. The transcoded videos, on the other hand, are the derived datasets that can be regenerated on-demand from their source videos. Therefore, they should only

TABLE I
SUMMARY OF CONCEPTS AND THEIR NOTATION

Notation	Description
τ_i	i^{th} transcoded video
NS_{τ_i}	new cost and popularity score of τ_i
RC_T	renting cost of a transcoding server per renting hour
S_{τ_i}	total cumulative cost and popularity score of τ_i
SC_{τ_i}	storage cost of τ_i in unit time
SC_m	monthly storage cost per 1 gigabytes
SD_{τ_i}	storage duration for transcoded video τ_i
TC_{τ_i}	transcoding cost of τ_i
TT_{τ_i}	transcoding time of τ_i
$VSmb_{\tau_i}$	transcoded video τ_i size in megabytes
GB_{mb}	megabytes to gigabytes conversion factor
H_{sec}	hour to seconds conversion factor
RP_S	month to desired time unit conversion factor

be stored in the *video repository* when it is cost-efficient to store them. Thus, the proposed strategy is only applicable to the transcoded videos. In other words, since the computation and the storage costs of the source videos are not relevant, the proposed strategy is based only on the computation and storage costs of the transcoded videos.

In cloud computing, the computation cost is essentially the cost of using VMs, which is usually calculated on an hourly basis. The storage cost, on the other hand, is often computed on a monthly basis. The computation cost of a transcoded video depends on its transcoding time and on how often the video is re-transcoded. Thus, if a video is frequently re-transcoded, the computation cost would increase rapidly. On the other hand, the storage cost of a transcoded video depends on the length of the storage duration and the video size on disk. Therefore, it increases gradually with the passage of time. The longer the duration, the higher the cost. Thus, our proposed strategy estimates an equilibrium point on the time axis where the computation cost and the storage cost of a transcoded video become equal. This estimated equilibrium point indicates the minimum duration for which the video should be stored in the *video repository*. Figure 2 shows that if a video is transcoded once and stored in the *video repository*, then initially the computation cost is higher than the storage cost. However, with the passage of time, the storage cost continues to increase until it becomes equal to the computation cost and then it grows even further unless the video is removed from the *video repository*. Thus, if the video is deleted before its estimated equilibrium point and then it is subsequently requested, the computation cost will increase due to the unnecessary re-transcoding. Likewise, if the video is stored beyond its estimated equilibrium point and then it does not receive a subsequent request, the storage cost will increase unnecessarily.

In an on-demand video streaming service, each transcoded video may be requested and viewed a number of times. Frequently viewed, popular videos get a lot of requests. While, sporadically viewed, less popular videos get only a few requests. For cost-efficient storage, it is essential to use an estimate of the popularity of the individual transcoded videos. This information can then be used to determine the

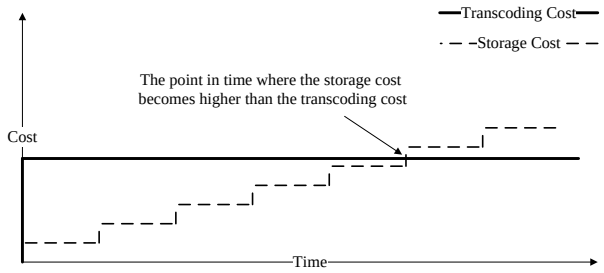

Fig. 2. The estimated equilibrium point between the storage cost and the transcoding cost of a transcoded video

exact duration for which a video should be stored in the *video repository*. Therefore, the proposed strategy accounts for the popularity of individual transcoded videos. It uses the estimated computation cost, the estimated storage cost, and the video popularity information to calculate a cost and popularity score S_{τ_i} for each transcoded video τ_i. The higher the score the longer the video is stored in the *video repository*. Thus, with the incorporation of the video cost and popularity score, it becomes justifiable to store popular transcoded videos beyond their estimated equilibrium point. In other words, it differentiates popular videos that should be stored for a longer duration.

In our proposed strategy, the storage cost SC_{τ_i} of a transcoded video τ_i is calculated as

$$SC_{\tau_i} = \frac{VSmb_{\tau_i}}{GB_{mb}} * \frac{SC_m}{RP_S} * SD_{\tau_i} \tag{1}$$

where $VSmb_{\tau_i}$ is the size of the transcoded video τ_i in megabytes, GB_{mb} is the megabytes to gigabytes conversion factor, SC_m is the monthly storage cost per 1 gigabytes of storage, RP_S is the month to desired time unit conversion factor, and SD_{τ_i} is the length of the storage duration for the transcoded video τ_i. Similarly, the transcoding cost TC_{τ_i} of a transcoded video τ_i is calculated as

$$TC_{\tau_i} = TT_{\tau_i} * \frac{RC_T}{H_{sec}} \tag{2}$$

where TT_{τ_i} is the transcoding time of τ_i, RC_T is the renting cost of a transcoding server per renting hour, and H_{sec} is the hour to seconds conversion factor, which is used to normalize the computation cost to a per second basis.

Whenever a new request for a transcoded video τ_i arrives at the *streaming server*, the video cost and popularity score S_{τ_i} is updated to reflect the new costs and the new popularity information. The new cost and popularity score NS_{τ_i} represents the estimated equilibrium point where the computation cost and the storage cost of τ_i become equal. Therefore, it indicates the minimum duration for which the video should be stored. The new cost and popularity score NS_{τ_i} of a video τ_i is calculated as the ratio of TC_{τ_i} and SC_{τ_i}

$$NS_{\tau_i} = \frac{TC_{\tau_i}}{SC_{\tau_i}} \tag{3}$$

Finally, the total cost and popularity score S_{τ_i} of a video τ_i is calculated by accumulating the new cost and popularity

score NS_{τ_i} of the said video over time. That is, for each new request of a transcoded video τ_i, we obtain the previous value of the total cost and popularity score S_{τ_i} of the transcoded video, calculate NS_{τ_i}, and then add them together to produce the new value of the S_{τ_i}. Moreover, the total cost and popularity score of a video that was not stored previously is set to NS_{τ_i}. The total cost and popularity score S_{τ_i} determines the exact duration for which a video τ_i should be stored.

Each transcoded video τ_i should be stored in the *video repository* for as long as it is cost-efficient to store it. However, when a video loses its popularity, it should be subsequently deleted to avoid unnecessary storage cost. Therefore, on certain time intervals, the proposed strategy performs the following steps for each transcoded video τ_i. It obtains the storage cost SC_{τ_i}, the cost and popularity score S_{τ_i}, and the transcoding cost TC_{τ_i}. Then, it multiplies S_{τ_i} and TC_{τ_i} and compares it with SC_{τ_i} as follows

$$SC_{\tau_i} > TC_{\tau_i} * S_{\tau_i} \tag{4}$$

If the inequality holds, it implies that it is cost-efficient to delete the transcoded video. Therefore, the video is removed from the *video repository*. However, if the inequality does not hold, it indicates that it is not cost-efficient to delete the video. Therefore, the video is not removed. Moreover, the cost and popularity score S_{τ_i} is decremented in accordance with the length of the time interval to reflect the passage of time. In this way, when a popular video loses its popularity, it starts losing its cost and popularity score as well until it is removed from the *video repository* or it gets some new requests to regain its popularity.

IV. Experimental Design and Setup

Software simulations are often used to test and evaluate new approaches and strategies involving complex environments [15], [16]. For our proposed strategy, we have developed a discrete-event simulation in the Python programming language. It is based on the SimPy simulation framework [17]. Also, for a comparison of results with the alternative existing approaches, we have developed a discrete-event simulation for two intuitive computation and storage trade-off strategies, which are the store all strategy and the usage based strategy [18]. The store all strategy stores all transcoded videos irrespective of their computation and storage costs. While the usage based strategy stores only popular videos and removes the rest. That is, it does not account for the computation and storage costs.

We considered two different load patterns in two separate experiments. Experiment 1 used a semisynthetic load pattern, while experiment 2 used a realistic load pattern, which was obtained from the real video access data provided by Bambuser AB[1].

For the computation and the storage costs, we used the Amazon Elastic Compute Cloud (EC2)[2] and the Amazon S3[3] cost models. The computation cost in Amazon EC2 is based on an hourly charge model. Whereas, the storage cost of Amazon S3 is based on a monthly charge model. In our experiments, we used only small instances. As of writing of this paper, the cost of a small instance in Amazon EC2 is $\$0.06$ per hour. The cost of storage space in Amazon S3 is based on a nonlinear cost model as shown in Table II.

TABLE II
Amazon S3 storage pricing

	Standard Storage
First 1 TB per month	$ 0.095 per GB
Next 49 TB per month	$ 0.080 per GB
Next 450 TB per month	$ 0.070 per GB
Next 500 TB per month	$ 0.065 per GB
Next 4000 TB per month	$ 0.060 per GB
Over 5000 TB per month	$ 0.055 per GB

The experiments used HD, SD (Standard-Definition), and mobile video streams. Since SD videos currently have a higher demand than the HD and mobile videos, we considered 20% HD, 30% mobile, and 50% SD video streams. The GOP size for different types of videos was different. For HD videos, the average size of a video segment was 75 frames with a standard deviation of 7 frames. Likewise, for SD and mobile videos, the average size of a segment was 250 frames with a standard deviation of 20 frames.

The transcoding rate depends on the video contents, such as, frame resolution, type of video format, type of frames, and contents of blocks. Different transcoding mechanisms also require different times. In our experiments, the average transcoding rate for SD videos was assumed to be four times of its play rate. As mobile videos have low resolution, the average transcoding rate for mobile videos was eight times the play rate. Since HD videos require more computation, the average transcoding rate for an HD video was assumed to be double of the play rate. In an on-demand video transcoding service, a source video is usually transcoded in many different formats. Therefore, we assumed that a source video can be transcoded into a maximum of 30 different formats. Likewise, since in an on-demand video streaming service, the number of source videos always continue to grow, we used a continuously increasing number of source videos in our experiments. However, since the number of the newly uploaded source videos is usually only a small fraction of the total number of downloaded videos, the video upload rate in our experiments was assumed to be 1% of the total number of the video download requests. In both experiments, the desired time unit for storage, as used in the month to desired time unit conversion factor RP_S, was assumed to be one day. Therefore, RP_S was 30. Moreover, the minimum storage duration for a transcoded video SD_{τ_i} was also assumed to be one day.

A. Experiment 1: Semisynthetic Load Pattern

The objective of experiment 1 was to evaluate the proposed strategy for larger videos spanning over multiple hours of play time. Therefore, the play time of a video in the semisynthetic

[1] http://bambuser.com/
[2] http://aws.amazon.com/ec2/
[3] http://aws.amazon.com/s3/

load pattern of experiment 1 was approximately from 1 hour to 2.5 hours. The load pattern consists of approximately 10 days of real video access data from Bambuser AB. However, it was stretched over a period of three months of the simulated time to limit the amount of memory required to run the simulations.

B. Experiment 2: Real Load Pattern

The objective of experiment 2 was to evaluate the proposed strategy for a realistic load pattern. Therefore, it used a real load pattern, which constitutes real video access data from Bambuser AB. The load pattern used in experiment 2 consists of approximately 30 days of real video access data. The total number of frames in a video stream was in the range of 18000 to 90000, which represents an approximate play time of 10 to 50 minutes with the frame rate of 30 frames per second.

V. Results and Analysis

In this section, we compare the experimental results of the proposed strategy with that of the store all strategy and the usage based strategy. Each result in Figure 3 to Figure 5 and Figure 7 to Figure 9 consists of seven different plots, which are number of user requests, number of transcoding servers, transcoding cost, storage cost, storage size, number of source videos, and number of transcoded videos. The number of user requests plot represents the load pattern of the video access data. In other words, it is the user load on the *streaming server*. Due to data confidentiality, the exact volume of the load can not be revealed. Therefore, we have omitted the scale of this plot from all the results. The number of transcoding servers plot shows the total number of *transcoding servers* being used at a particular time. The transcoding cost plot represents the total computation cost of all transcoded videos in US dollars. Similarly, the storage cost plot shows the storage cost in US dollars of all transcoded videos, which are stored in the *video repository*. The storage size plot represents the total size of the cloud storage used to store the transcoded videos. The number of source videos plot shows the total number of source videos in the *video repository*. Likewise, the number of transcoded videos is the total number of transcoded videos in the *video repository*.

A. Experiment 1: Semisynthetic Load Pattern

Figure 3 presents the simulation results of the store all strategy from experiment 1. The results span over a period of three months as represented by the number of user requests plot. The total number of transcoded videos was 18206, while the total number of source videos was 4571. The total transcoding cost was $2882.7, the total storage cost was $3202.56, and the total storage size was 17.25 terabytes. Since the store all strategy stores all transcoded videos irrespective of their computation and storage costs, the storage cost was very high due to a large number of transcoded videos stored in the *video repository*. Therefore, the results indicate that the store all strategy is not cost-efficient.

Figure 4 presents the results of the usage based strategy from experiment 1. The total number of transcoded videos

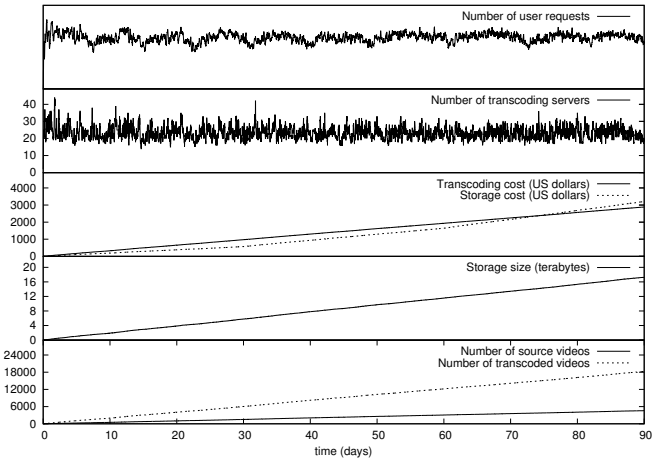

Fig. 3. Experiment 1: store all strategy

was 17955 for the same number of source videos as used in the store all strategy. The total transcoding cost was $2735.76, the total storage cost was $2905.92, and the total storage size was 14.91 terabytes. Since the usage based strategy stores only popular videos, the storage cost of the usage based strategy was slightly less than that of the store all strategy. Therefore, the results indicate that the usage based strategy is cost-efficient when compared to the store all strategy. However, since it does not account for the computation and the storage costs, it may remove some videos that have a high transcoding cost.

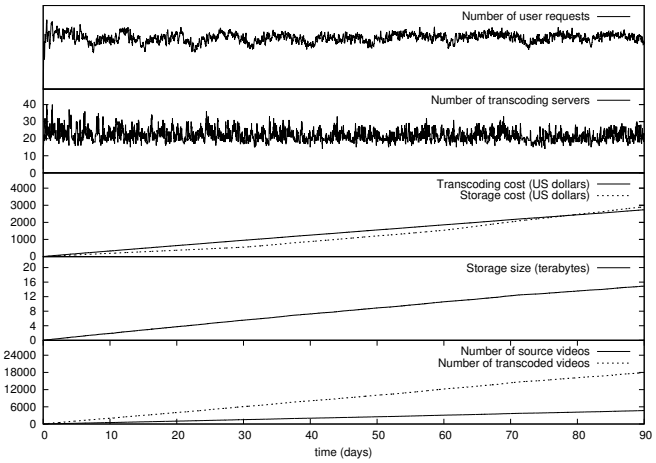

Fig. 4. Experiment 1: usage based strategy

Figure 5 presents the results of the proposed strategy from experiment 1. The total number of transcoded videos was 5226 for the same number of source videos as used in the store all strategy and the usage based strategy. The total transcoding cost was $3221.04, the total storage cost was $1592.64, and the storage size was 7.05 terabytes. Since the proposed strategy accounts for the computation cost, the storage cost, and the video popularity information, the storage cost was much less than that of the store all strategy and the usage based strategy.

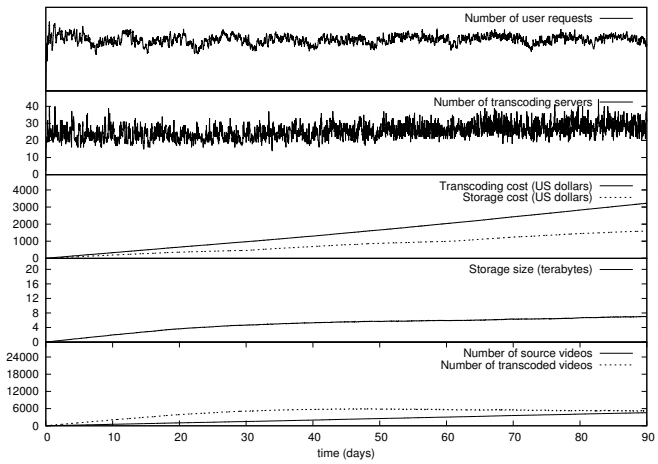

Fig. 5. Experiment 1: cost and popularity score based strategy

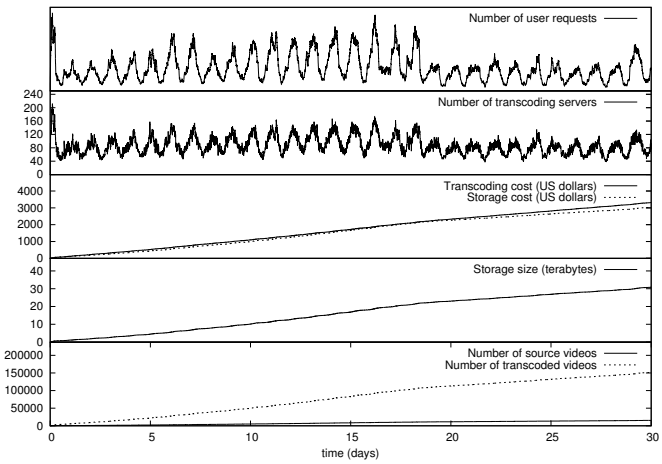

Fig. 7. Experiment 2: store all strategy

Figure 6 presents a comparison of the total costs in experiment 1. The total cost consists of the computation cost and the storage cost. The results show that the store all strategy has the highest total cost. The usage based strategy has slightly less total cost than the store all strategy. Moreover, the proposed storage has the least total cost among all the three strategies. Therefore, experiment 1 results indicate that the proposed strategy is cost-efficient when compared to the store all and the usage based strategies.

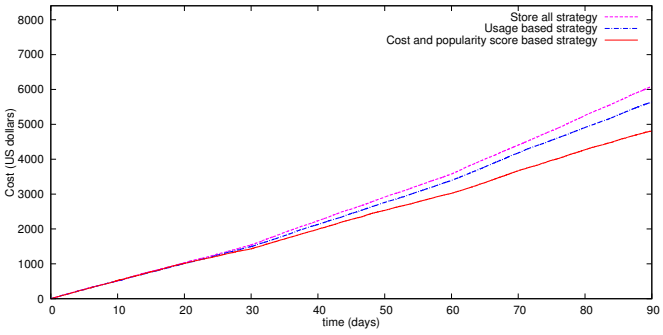

Fig. 6. Experiment 1: total cost comparison

B. Experiment 2: Real Load Pattern

Figure 7 presents the simulation results of the store all strategy from experiment 2. The results span over a period of one month as represented by the number of user requests plot. The total number of transcoded videos was 151564, while the total number of source videos was 15340. The total transcoding cost was $3314.7, the total storage cost was $3031.68, and the total storage size was 30.83 terabytes. As in experiment 1, the storage cost in experiment 2 was also very high due to a large number of transcoded videos stored in the *video repository*. Therefore, experiment 2 results also indicate that the store all strategy is not cost-efficient.

Figure 8 presents the results of the usage based strategy from experiment 2. The total number of transcoded videos was 147610 for the same number of source videos as used in the store all strategy. The total transcoding cost was $3155.58, the total storage cost was $2690.88, and the total storage size was 27.37 terabytes. As in experiment 1, the storage cost of the usage based strategy in experiment 2 was also slightly less than that of the store all strategy. Therefore, experiment 2 results also indicate that the usage based strategy is cost-efficient when compared to the store all strategy.

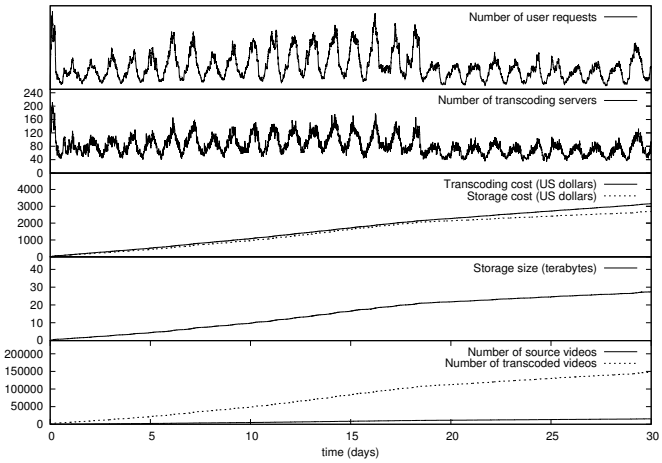

Fig. 8. Experiment 2: usage based strategy

Figure 9 presents the results of the proposed strategy from experiment 2. The total number of transcoded videos was 55646 for the same number of source videos as used in the store all strategy and the usage based strategy. The total transcoding cost was $3515.82, the total storage cost was $1584.96, and the total storage size was 12.21 terabytes. Since the proposed strategy accounts for the computation cost, the storage cost, and the video popularity information, the storage cost was much less than that of the store all strategy and the usage based strategy.

Figure 10 presents a comparison of the total costs in experiment 2, which consists of the computation cost and the

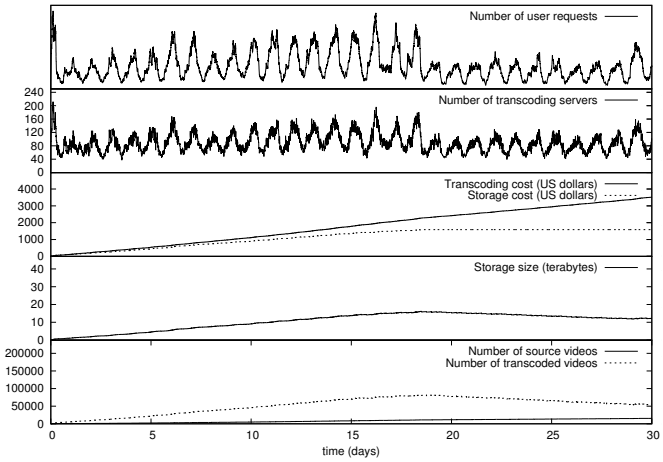

Fig. 9. Experiment 2: cost and popularity score based strategy

storage cost. The results are similar to the results of experiment 1. Again, the store all strategy has the highest total cost. The usage based strategy has slightly less total cost than the store all strategy. Moreover, the proposed strategy has the least total cost among all the three strategies. Therefore, experiment 2 results also indicate that the proposed strategy is cost-efficient when compared to the store all and the usage based strategies.

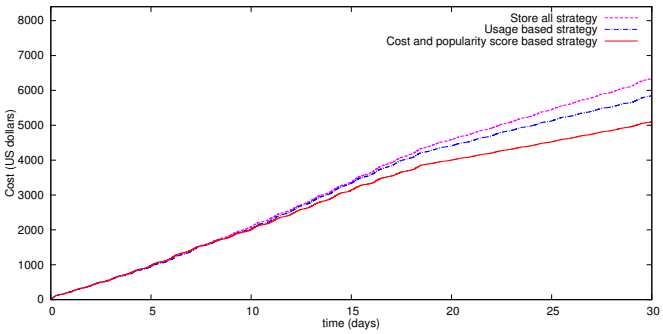

Fig. 10. Experiment 2: cost comparison

VI. Related Work

There are currently only a few works in the area of computation and storage trade-off analysis for cost-efficient usage of cloud resources. One of the earlier attempts include Adams et al. [19], who addressed the problem of maximizing efficiency by trading storage for computation. They highlighted some of the important issues and factors involved in constructing a cost-benefit model, which can be used to analyze the trade-offs between computation and storage. However, they did not propose a strategy to find the right balance between computation and storage resources.

Deelman et al. [20] studied cost and performance trade-offs for an astronomy application using Amazon EC2 and Amazon S3 cost models. They also examined the trade-offs between three different data management models for cloud storage. The authors concluded that, based on the likelihood of reuse, storing popular datasets in the cloud can be cost-effective. However, they did not provide a concrete strategy for cost-effective computation and storage of scientific datasets in the cloud.

Nectar system [21] is designed to automate the management of data and computation in a data center. It initially stores all the derived datasets when they are generated. However, when the available disk space falls below a threshold, all obsolete or least valued datasets are garbage collected to improve resource utilization. Nectar makes use of the usage history of datasets to perform cost-benefit analysis, which determines the usefulness of each dataset. The cost-benefit analysis considers the size of the dataset, the elapsed time since it was last used, the number of times it has been used, and its cumulative computation time. The datasets with the largest cost-to-benefit ratios are deleted. Although Nectar provides a computation and storage trade-off strategy, it is not designed to reduce the total cost of computation and storage in a cloud-based service, which uses IaaS resources.

Yuan et al. [18] proposed two strategies for cost-effective storage of scientific datasets in the cloud, which compare the computation cost and the storage cost of the datasets. They also presented a Cost Transitive Tournament Shortest Path (CTT-SP) algorithm to find the best trade-off between the computation and the storage resources. Their strategies are called cost rate based storage strategy [22], [23] and local-optimization based storage strategy [24]. The cost rate based storage strategy compares computation cost rate and storage cost rate to decide storage status of a dataset. Whereas, the local-optimization based storage strategy partitions a data dependency graph (DDG) of datasets into linear segments and applies the CTT-SP algorithm to achieve a localized optimization. The local-optimization based storage strategy tends to be more cost-effective than the cost rate based storage strategy. However, due to the overhead introduced by the CTT-SP algorithm, it is less efficient and less scalable. In contrast to the cost rate based storage strategy [22], [23], our proposed trade-off strategy estimates an equilibrium point on the time axis where the computation cost and the storage cost of a transcoded video become equal. Moreover, it estimates video popularity of the individual transcoded videos to differentiate popular videos. The DDG-based local-optimization based storage strategy of Yuan et al. [24], which provides cost-effective results for scientific datasets, is not much relevant for video transcoding because video transcoding does not involve a lot of data dependencies.

Most of the existing computation and storage trade-off strategies described above were originally proposed for scientific datasets. To the best of our limited knowledge, there are currently no existing computation and storage trade-off strategies for video transcoding. The difference of application domain may play a vital role when determining cost-efficiency of the existing strategies. Therefore, some of the existing strategies may have limited efficacy and little cost-efficiency for video transcoding.

In addition to the above mentioned works, there are also a

few works on some of the related topics, such as, a hybrid scheme to determine an optimal threshold between static transcoding and dynamic transcoding [25], video transcoding service in cloud computing [26], video segmentation for distributed video transcoding [12], [13], dynamic resource allocation for video transcoding [11], and admission control and scheduling for video transcoding in the cloud [16].

VII. Conclusion

In this paper, we proposed a cost-efficient computation and storage trade-off strategy for video transcoding in the cloud. The proposed strategy estimates the computation cost, the storage cost, and the video popularity information of individual transcoded videos and then uses 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. The objective is to reduce the total IaaS cost by trading storage for computation, or vice versa. We presented a discrete-event simulation of the proposed strategy along with an experimental evaluation involving semisynthetic and realistic load patterns. Also, for the sake of comparison, we simulated two intuitive computation and storage trade-off strategies and compared their results with that of the proposed strategy. The results indicate that our proposed strategy is more cost-efficient than the two intuitive strategies. It provided a good trade-off between the computation resources and the storage resources for the semisynthetic as well as realistic load patterns.

Acknowledgements

This work was supported by the Cloud Software Finland research project and by an Amazon Web Services research grant. Fareed Jokhio and Adnan Ashraf were partially supported by the Foundation of Nokia Corporation and by doctoral scholarships from the Higher Education Commission (HEC) of Pakistan. The authors want to thank Bambuser AB for providing real video access data to perform experiments.

References

[1] J. Watkinson, *The MPEG Handbook: MPEG-1, MPEG-2, MPEG-4*, ser. Broadcasting and communications. Elsevier/Focal Press, 2004.

[2] T. Wiegand, G. J. Sullivan, and A. Luthra, "Draft ITU-T recommendation and final draft international standard of joint video specification," in *Technical Report*, 2003.

[3] H. Sun, W. Kwok, and J. Zdepski, "Architectures for mpeg compressed bitstream scaling," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 6, no. 2, pp. 191 –199, apr 1996.

[4] K.-S. Kan and K.-C. Fan, "Video transcoding architecture with minimum buffer requirement for compressed mpeg-2 bitstream." *Signal Processing*, vol. 67, no. 2, pp. 223–235, 1998.

[5] P. A. Assuncao and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of mpeg-2 bit streams," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 8, no. 8, pp. 953–967, Dec. 1998. [Online]. Available: http://dx.doi.org/10.1109/76.736724

[6] G. Keesman, R. Hellinghuizen, F. Hoeksema, and G. Heideman, "Transcoding of MPEG bitstreams," *Signal Processing: Image Communication*, vol. 8, no. 6, pp. 480–500, 1996.

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

[8] Y.-P. Tan and H. Sun, "Fast motion re-estimation for arbitrary downsizing video transcoding using h.264/avc standard," *IEEE Trans. on Consum. Electron.*, vol. 50, no. 3, pp. 887–894, 2004.

[9] G. Shen, Y. He, W. Cao, and S. Li, "MPEG-2 to WMV transcoder with adaptive error compensation and dynamic switches," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 16, no. 12, pp. 1460–1476, Dec. 2006.

[10] H. Kato, Y. Takishima, and Y. Nakajima, "A fast DV to MPEG-4 transcoder integrated with resolution conversion and quantization," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 17, no. 1, pp. 111–119, Jan. 2007.

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

[12] F. Jokhio, T. Deneke, S. Lafond, and J. Lilius, "Bit rate reduction video transcoding with distributed computing," in *Parallel, Distributed and Network-Based Processing (PDP), 2012 20th Euromicro International Conference on*, feb. 2012, pp. 206 –212.

[13] F. A. Jokhio, T. Deneke, S. Lafond, and J. Lilius, "Analysis of video segmentation for spatial resolution reduction video transcoding," in *Intelligent Signal Processing and Communications Systems (ISPACS), 2011 International Symposiuml*, Dec 2011, p. 6 pp.

[14] A. Ashraf, B. Byholm, and I. Porres, "A session-based adaptive admission control approach for virtualized application servers," in *Utility and Cloud Computing (UCC), 5th IEEE/ACM International Conference on*, 2012, pp. 65–72.

[15] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. 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, no. 1, 2011.

[16] 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 Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pp. 482–489.

[17] N. Matloff, *A Discrete-Event Simulation Course Based on the SimPy Language*. University of California at Davis, 2006.

[18] D. Yuan, Y. Yang, X. Liu, and J. Chen, "Computation and storage trade-off for cost-effectively storing scientific datasets in the cloud," in *Handbook of Data Intensive Computing*, B. Furht and A. Escalante, Eds. Springer New York, 2011, pp. 129–153.

[19] I. F. Adams, D. D. E. Long, E. L. Miller, S. Pasupathy, and M. W. Storer, "Maximizing efficiency by trading storage for computation," in *Proceedings of the 2009 conference on Hot topics in cloud computing*, ser. HotCloud'09. Berkeley, CA, USA: USENIX Association, 2009.

[20] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, ser. SC '08. Piscataway, NJ, USA: IEEE Press, 2008, pp. 50:1–50:12.

[21] P. K. Gunda, L. Ravindranath, C. A. Thekkath, Y. Yu, and L. Zhuang, "Nectar: automatic management of data and computation in datacenters," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–8.

[22] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A cost-effective strategy for intermediate data storage in scientific cloud workflow systems," in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, 2010, pp. 1–12.

[23] D. Yuan, Y. Yang, X. Liu, G. Zhang, and J. Chen, "A data dependency based strategy for intermediate data storage in scientific cloud workflow systems," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 9, pp. 956–976, 2012.

[24] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A local-optimisation based strategy for cost-effective datasets storage of scientific applications in the cloud," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, 2011, pp. 179–186.

[25] I. Shin and K. Koh, "Hybrid transcoding for QoS adaptive video-on-demand services," *IEEE Trans. on Consum. Electron.*, vol. 50, no. 2, pp. 732–736, May 2004.

[26] 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 *22nd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2012.