Stream-Based Admission Control and Scheduling for Video Transcoding in Cloud Computing

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

***Abstract*—This paper presents a novel approach for stream-based admission control and job scheduling for video transcoding called SBACS (Stream-Based Admission Control and Scheduling). SBACS uses queue waiting time of transcoding servers to make admission control decisions for incoming video streams. It implements stream-based admission control with per stream admission. To ensure efficient utilization of the transcoding servers, video streams are segmented at the *Group of Pictures* level. In addition to the traditional rejection policy, SBACS also provides a stream deferment policy, which exploits cloud elasticity to allow temporary deferment of the incoming video streams. In other words, the admission controller can decide to admit, defer, or reject an incoming stream and hence reduce rejection rate. In order to prevent *transcoding jitters* in the admitted streams, we introduce a job scheduling mechanism, which drops a small proportion of video frames from a video segment to ensure continued delivery of video contents to the user. The approach is demonstrated in a discrete-event simulation with a series of experiments involving different load patterns and stream arrival rates.**

***Index Terms*—Video transcoding; cloud computing; admission control; scheduling; segmentation; resource allocation**

I. Introduction

Streaming of digital videos is increasingly common among the Internet users. Video streaming of a large number of videos may require a lot of resources on the server-side. Therefore, for efficient use of storage and transmission media, digital videos are often stored and transmitted in compressed formats, such as MPEG-4 [1] and H.264 [2]. Client-side devices that are used to stream and play videos usually support only a subset of the existing video formats. A video on the server-side may be stored in a different format than those supported by a target device. Therefore, the video must be converted into a format that is supported by the target device [3]. The process of converting a video from one compressed format into another compressed format is known as video transcoding [4]. It involves re-encoding according to new requirements of frame resolution, bit rate, frame rate, and video format.

Video transcoding for an on-demand video streaming service needs to be done on-the-fly in realtime [5]. Since video transcoding is a compute-intensive operation, transcoding of a large number of video streams requires a large-scale cluster-based distributed system. Infrastructure as a Service (IaaS) clouds, such as Amazon Elastic Compute Cloud (EC2) [6], provide all necessary resources for creating a dynamically scalable tier of transcoding servers. In our previous work [7], we presented a prediction-based dynamic resource allocation approach to scale video transcoding service on a given IaaS cloud. However, resource allocation alone does not prevent servers from becoming overloaded [8]. Therefore, resource allocation should be augmented with an admission control mechanism with the goal to prevent servers from becoming overloaded by restricting incoming load on them.

In this paper, we present the SBACS (Stream-Based Admission Control and Scheduling) approach for a dynamically scalable tier of video transcoding servers. It provides video stream-based admission control on a per stream level. SBACS uses queue waiting time of transcoding servers to make admission control decisions. For preemptive control, it uses a two-step load prediction approach [9], which predicts queue waiting times on individual servers. In addition to the traditional load rejection policy, SBACS also provides a stream deferment policy, which allows to temporarily defer an arrived stream until a new virtual machine (VM) is provisioned to start another transcoding server or an existing server becomes less loaded. To ensure efficient utilization of the transcoding servers, server resources are shared among admitted streams by performing video segmentation at the *Group of Pictures (GOP)* level. The video segments are then sent to the servers via a load balancer. SBACS also provides a job scheduling algorithm, which aims to prevent *transcoding jitters* in the admitted streams by dropping a small proportion of video frames to ensure continued delivery of video contents to the users. We proceed as follows. Section II discusses important related works. Section III outlines the main tasks and the most important characteristics of the proposed approach. Section IV presents the system architecture. The proposed admission control and job scheduling algorithms are described in Section V. In Section VI, we present simulation results before concluding in Section VII.

II. Related Work

The existing works on admission control for web-based systems can be classified according to the scheme presented in [10]. For instance, [11] and [12] are control-theoretic approaches, while [13] and [14] use machine learning techniques.

Similarly, [15], [10], [16], [17], and [8] are utility-based approaches.

Cherkasova and Phaal [15] proposed an admission control approach that uses the traditional *on-off* control. It measures server loads during predefined time intervals and uses them to compute the new predicted load for the next interval. If the predicted load exceeds certain threshold, the admission controller rejects new incoming load in the next time interval and serves only the already admitted users. Once the predicted load drops below its threshold, the server changes its policy for the next time interval and begins to accept new load again. Almeida et al. [10] presented a joint resource allocation and admission control approach for a virtualized platform hosting a number of web applications, where each VM runs a dedicated web service application. The optimization objective is to maximize the provider's revenue, while satisfying the customers' Quality of Service (QoS) requirements and minimizing the cost of resource utilization. The approach dynamically adjusts the fraction of capacity assigned to each VM and limits the incoming workload by serving only the subset of requests that maximize profits.

Chen et al. [16] proposed admission control based on estimation of service times (ACES). That is, to differentiate and admit requests based on the amount of processing time required by a request. In ACES, admission of a request is decided by comparing the available computation capacity to the predetermined delay bound of the request. Shaaban and Hillston [17] presented cost-based admission control (CBAC), which uses a congestion control technique. Rather than rejecting user requests at high load, CBAC uses a discount-charge model to encourage users to postpone their requests to less loaded time periods. However, if a user chooses to proceed with the request in a high load period, then an extra charge is imposed. The model is suitable only for e-commerce web sites when more users place orders that involve monetary transactions. Ashraf et al. [8] proposed a session-based adaptive admission control approach for virtualized application servers called ACVAS. It uses per session admission control [13] with a load deferment mechanism to reduce the number of rejected sessions.

Muppala and Zhou [13] proposed the coordinated session-based admission control approach (CoSAC), which provides admission control for multi-tier web applications with per session admission control. CoSAC also provides coordination among the states of tiers with a machine learning technique using a Bayesian network. Huang et al. [14] proposed admission control schemes for proportional differentiated services. The paper proposes two admission control schemes to enable proportional delay differentiated service (PDDS) at the application level. Each scheme is augmented with a prediction mechanism, which predicts the total maximum arrival rate and the maximum average waiting time for each priority class. When a user request belonging to a specific priority class arrives, the admission control algorithm uses the time series predictor to forecast the average arrival rate of the class for the next interval, computes the average waiting time for the class for the next interval, and determines if the incoming user request should be admitted.

Voigt and Gunningberg [11] proposed admission control based on the expected resource consumption of the requests, including a mechanism for service differentiation that guarantees low response time and high throughput for premium clients. The approach avoids over-utilization of individual server resources, which are protected by dynamically setting the acceptance rate of resource-intensive requests. The adaptation of the acceptance rates (average number of requests per second) is done by using proportional-derivative (PD) feedback control loops. Robertsson et al. [12] proposed an admission control mechanism for a web server system with control-theoretic methods. It uses a control-theoretic model of a G/G/1 system with an admission control mechanism for nonlinear analysis and design of controller parameters for a discrete-time proportional-integral (PI) controller. The controller calculates the desired admittance rate based on the reference value of average server utilization and the estimated or measured load situation. It then rejects those requests that could not be admitted.

All existing admission control approaches described above were originally proposed for web applications. To the best of our limited knowledge, there are no existing admission control approaches for video transcoding servers. However, there are a few research works on some of the related topics, such as, video transcoding service in cloud computing [18], video segmentation for distributed video transcoding [19], [20], and dynamic resource allocation for video transcoding [7].

III. SBACS Approach

The main tasks of the proposed approach are to make admission control and job scheduling decisions for a scalable tier of transcoding servers, in which each server runs on a virtual machine. The most important characteristics of the proposed approach are as follows.

A. Stream-Based Admission Control with Per Stream Admission

Cherkasova and Phaal [15] used the traditional *on-off* control in their admission control approach. In *on-off* control, acceptance of new user load is turned on or off for an entire admission control interval. The admission control decisions are made at the interval boundaries and can not be changed inside an interval. Therefore, *on-off* control may lead to over-admission, especially when handling a bursty load, which can result in overloading of servers. Some of the recent admission control approaches for web applications, such as [13], [8], use per session admission control. SBACS uses a similar approach for video streams. It implements a per stream admission control, which reduces over-admission by making an admission control decision for each incoming stream.

B. Stream Deferment Mechanism

Most of the traditional admission control approaches rely only on load rejection policy to prevent server overloading.

SBACS uses a simple mechanism to defer new video streams that would otherwise be rejected [8]. Such streams are deferred on an *entertainment server* until a new server is provisioned or an existing server becomes less loaded. However, if the *entertainment server* also approaches its capacity limits, the new streams are rejected. Therefore, for each new video stream, the *admission controller* makes one of the three possible decisions: admit the stream, defer the stream, or reject the stream.

C. Job Scheduling Based on Queue Waiting Time

One of the main characteristics of the proposed approach is that in addition to an admission control mechanism, it also features a job scheduling algorithm. The algorithm uses queue waiting time of individual transcoding servers to further prevent servers from becoming overloaded and to prevent *transcoding jitters* in the admitted video streams. For overload prevention on a sufficiently utilized server, it starts dropping a small proportion of video frames from each subsequent transcoding segment on the server. Likewise, for preventing *jitters* in a video stream, it computes the estimated delivery deadline, the estimated transcoding time, and the estimated response time of each video segment. If it finds a deadline violation, the violation is prevented by dropping some video frames in proportion to the degree of violation.

D. Load Prediction Models

Existing admission control approaches, such as [13], [14], [15], [17], [8], use a prediction of future load to improve admission control decisions by acting preemptively. Cherkasova and Phaal [15] computed predicted load by assigning certain weights to the current and the past loads. Muppala and Zhou [13] used the *exponential moving average* (EMA) method. Huang et al. [14] used machine learning techniques called support vector regression (SVR) and particle swarm optimization (PSO) for time-series prediction. Shaaban and Hillston [17] assumed a repeating pattern of workload over a suitable time period.

For efficient runtime decision making, it is essential to avoid prediction models that require intensive computation, frequent updates to their parameters, or (off-line) training. Ashraf et al. [8] extended and used a two-step load prediction approach [9], which has been designed to predict future resource loads under realtime constraints. The two-step approach consists of a load tracker and a load predictor. SBACS also uses the extended two-step load prediction approach to predict queue waiting time of individual transcoding servers. It uses EMA for the load tracker and a simple linear regression model [21] for the load predictor.

E. Criterion for Assessment of Admission Control Efficiency

The efficiency of an admission control mechanism may be assessed in a number of different ways. Traditional admission control approaches that are designed for a fixed number of servers may be evaluated based on server overload prevention and increase in the session throughput. Likewise, Cherkasova and Phaal [15] used a QoS metric based on the number of aborted and rejected connections. However, for dynamically scalable server tiers, the efficiency of an admission control approach should be based on the tradeoff between cost and QoS. Therefore, Ashraf et al. [8] proposed a criterion for the assessment of admission control efficiency based on the tradeoff between the number of servers and six important QoS metrics. SBACS adopts it for video transcoding servers by defining QoS in terms of zero overloaded servers, maximum achievable throughput, minimum deferred streams, zero rejected streams, minimum *transcoding jitters*, and minimum dropped frames.

IV. System Architecture

The system architecture of the cloud-based video transcoding service is shown in Figure 1. It consists of a *streaming server*, a stream *splitter*, a stream *merger*, a *video repository*, a dynamically scalable cluster of *transcoding servers*, a *load balancer*, a *master controller and resource allocator*, a *load predictor*, an *admission controller*, and an *entertainment server*.

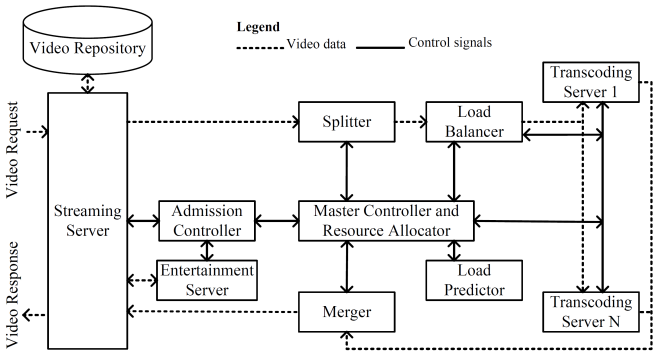

Fig. 1. System architecture

The *streaming server* accepts video requests from users and checks if the required video is found in the *video repository*. If it finds the video in the desired format, 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 *merger*, it starts streaming the video. To avoid unnecessary repetition of transcoding operations, we store a copy of each transcoded video in the *video repository* for a certain amount of time, typically a few days.

The *load balancer* distributes load on the *transcoding servers*. It implements the shortest queue waiting time policy, which selects a server with the least waiting time. The *master controller and resource allocator* provisions and releases VMs from the cluster of *transcoding servers*. The resource allocation and deallocaiton is mainly based on the target play rate of video streams and the predicted transcoding rate of transcoding servers. Our resource allocation and load prediction algorithms are described in detail in [7]. In this paper, our primary focus is on admission control and job scheduling algorithms, which are presented in Section V.

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). Two consecutive frames of a video often have small differences. Therefore, a frame can be stored using less bits once inter-frame redundancy is removed. Figure 2 shows a compressed MPEG-4 video stream consisting of different types of frames. *I*-frames are key frames and they do not use any other frames as a reference frame in the transcoding process. A *P*-frame requires a past frame as a reference frame. *B*-frames require both past and future frames as reference frames [1]. An *I*-frame followed by *B* and *P* frames is termed as a *GOP*. *GOPs* represent atomic units that can be transcoded independently of one another [7].

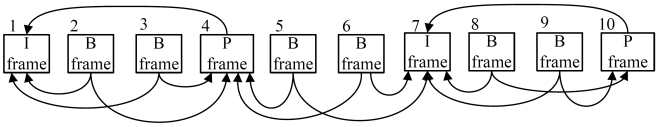

Fig. 2. A compressed MPEG-4 video stream

The *splitter* is responsible for video segmentation at the *GOP* level, while the *merger* merges the transcoded segments. The actual transcoding is performed by the *transcoding servers*. They get compressed video segments, perform the required transcoding operations, and return back the transcoded segments for merging. The *entertainment server* is used to temporarily defer new video streams that would otherwise be rejected.

V. Admission Control and Job Scheduling

In this section, we present the proposed admission control and job scheduling algorithms. For the sake of clarity, we provide the notations used in this section in Table I.

TABLE I
SUMMARY OF CONCEPTS AND THEIR NOTATION

DF_j	degradation factor of transcoding job j
DV_j	degree of violation of job j
ED_j	estimated deadline of job j
ER_j	estimated response time of job j
ET_j	estimated transcoding time of job j
j	transcoding job j
$\hat{j}$	job j after dropping frames
LA_{ent}	load average of entertainment server
MU_{ent}	memory utilization of entertainment server
NF_j	number of frames in job j
$\hat{NF}_j$	number of frames in job j after dropping frames
Q_{time_s}	actual queue waiting time of server s
$Q_{\hat{time}_s}$	predicted queue waiting time of server s
S	set of transcoding servers
S_{open}	set of open transcoding servers
$Stream_d$	set of deferred video streams
$Stream_n$	set of newly arrived video streams
LA_{UT}	load average upper threshold
MU_{UT}	memory utilization upper threshold
$Q_{time_{LT}}$	queue waiting time lower threshold
$Q_{time_{UT}}$	queue waiting time upper threshold
$admit(st)$	admit stream st
$defer(st)$	defer stream st
$dropF(j)$	calculate DV_j and drop frames
$pop(list)$	remove and return first element of the $list$
$reject(st)$	reject stream st
$schedule(j)$	schedule job j

A. Admission Control

The main task of our SBACS approach is to make admission control decisions for transcoding requests in a dynamically scalable tier of transcoding servers, which consists of virtualized servers. Since provisioning of a VM from a contemporary IaaS provider is not instantaneous [8], the proposed admission control approach provides a stream deferment mechanism to handle VM provisioning delays.

The admission control decisions are based on the states of the transcoding servers. A server s is considered *open* for existing as well as new incoming streams if its queue waiting time Q_{time_s} remains below a predefined lower threshold of the queue waiting time $Q_{time_{LT}}$. However, if Q_{time_s} exceeds $Q_{time_{LT}}$, while it still remains below the upper threshold of the queue waiting time $Q_{time_{UT}}$, the server is considered *closed* for the new incoming streams. A *closed* server is a sufficiently utilized server, which may become overloaded if the admission controller continues to admit more streams. Likewise, if Q_{time_s} also exceeds $Q_{time_{UT}}$, the server is considered *overloaded*. *Overloaded* is an undesirable state, which is characterized by very high waiting times and low server throughput.

Algorithm 1 presents our admission control algorithm. The algorithm is activated when a new video transcoding request arrives at the admission controller or when it finds at least one deferred transcoding request (line 2). For each new request, the admission controller makes one of the three possible decisions: admit the request, defer the request, or reject the request. All requests are served on the First Come First Served (FCFS) basis. However, to prevent deferred streams from starvation, deferred streams are given priority over new streams. Therefore, if the algorithm finds at least one *open* server (line 4) and at least one deferred stream (line 5), it admits a deferred stream (line 6). Otherwise, if there are no deferred streams, it admits a new stream (line 8). However, if it does not find an *open* server, it defers or rejects the new streams based on the state of the *entertainment server*. That is, if the *entertainment server* can accommodate more deferred streams, the new streams are deferred (line 12). For each deferred stream, the *entertainment server* sends a wait message to the user. However, if the *entertainment server* also reaches its capacity limits, the new streams are rejected (line 14).

B. Job Scheduling

As described in Section III-C, SBACS also provides a job scheduling algorithm to prevent *transcoding jitters* in the admitted streams. The job scheduling algorithm also complements the proposed admission control approach in preventing overloading of the transcoding servers. Algorithm 2 presents our job scheduling algorithm. For jitter prevention, it computes estimated delivery deadline ED_j, estimated transcoding time

Algorithm 1 Admission Control

1: **while true do**
2: **if** $|Stream_d| \geq 1 \vee |Stream_n| \geq 1$ **then**
3: $S_{open} := \{\forall s \epsilon S | Q_{\hat{time_s}} < Q_{time_{LT}}\}$
4: **if** $|S_{open}| \geq 1$ **then**
5: **if** $|Stream_d| \geq 1$ **then**
6: $admit(pop(Stream_d))$
7: **else**
8: $admit(pop(Stream_n))$
9: **end if**
10: **else if** $|Stream_n| \geq 1$ **then**
11: **if** $LA_{ent} < LA_{UT} \wedge MU_{ent} < MU_{UT}$ **then**
12: $defer(pop(Stream_n))$
13: **else**
14: $reject(pop(Stream_n))$
15: **end if**
16: **end if**
17: **end if**
18: **end while**

ET_j, and estimated response time ER_j of each transcoding job j (line 3). If it finds a deadline violation (line 4), it tries to prevent the violation by dropping some video frames in proportion to the degree of violation DV_j (line 5), which is computed by adding the current clock time $current_{time}$ and ER_j and then subtracting ED_j

$$DV_j = ER_j + current_{time} - ED_j \tag{1}$$

Dropping of frames in a video is termed as temporal resolution reduction [4]. Figure 3 shows a video segment before and after applying temporal resolution reduction. The main benefit of temporal resolution reduction is that the transcoding time of the video segment can be significantly reduced without greatly compromising the video quality. Moreover, the computational overheads of temporal resolution reduction are negligible, especially when dropping B-frames.

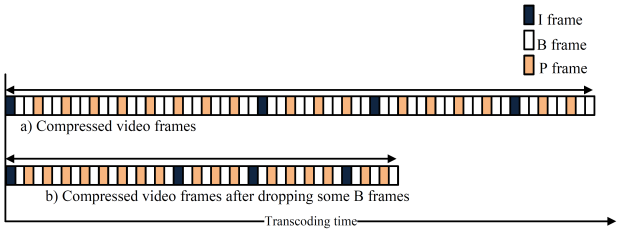

Fig. 3. Temporal resolution reduction

The temporal resolution reduction is applied at the *GOP* level, which consists of a certain number of frames. If a *GOP* had NF_j frames before applying temporal resolution reduction and it has $\hat{NF}_j$ frames afterwards, the video degradation factor of job j, DF_j, can be computed as the ratio of the dropped frames and NF_j

$$DF_j = \frac{NF_j - \hat{NF}_j}{NF_j} \tag{2}$$

Algorithm 2 Job Scheduling

1: **for** each job j arrived on a transcoding server s **do**
2: get ED_j, ET_j
3: $ER_j = ET_j + Q_{time_s}$
4: **if** ER_j violates ED_j **then**
5: $\hat{j} = dropF(j)$
6: $schedule(\hat{j})$
7: **else**
8: $schedule(j)$
9: **end if**
10: **if** $Q_{time_s} > Q_{time_{LT}}$ **then**
11: $\hat{j} = dropF(j)$
12: $schedule(\hat{j})$
13: **end if**
14: **end for**

A higher number of dropped frames results in a higher value of DF_j, while a reduced number of dropped frames results in a lower value of DF_j. Therefore, our job scheduling approach drops a reduced number of frames that satisfies estimated deadline ED_j.

The transcoding times of different types of video frames vary from one another as shown in Figure 4. The transcoding time of an *I*-frame is usually greater than that of a *P*-frame, which is greater than the transcoding time of a *B*-frame. The estimated transcoding time ET_j of a *GOP* containing NF_j frames is the sum of the estimated transcoding times of individual frames

$$ET_j = \alpha * ET_i + \beta * ET_p + \gamma * ET_b \tag{3}$$

where ET_i, ET_p, and ET_b represent estimated transcoding time of an *I*, a *P*, and a *B* frame, while α, β and γ denote number of frames of *I*, *P*, and *B* types respectively.

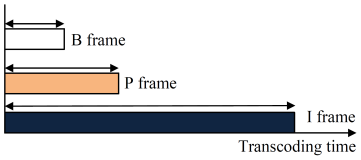

Fig. 4. Transcoding time of different types of frames

To complement overload prevention, the job scheduling algorithm starts dropping a small proportion of video frames from each subsequent transcoding segment on a *closed* server (line 11). Therefore, the server can be prevented from becoming overloaded by reducing the amount of required computation and the estimated transcoding time ET_j of each new transcoding segment.

VI. Simulation Results

Software simulations are often used to test and evaluate new algorithms involving complex environments [22]. We have developed a discrete-event simulation for SBACS. The simulation is written in the Python programming language and is based on the SimPy simulation framework [23].

A. Experimental Design and Setup

We considered two different synthetic load patterns in two separate experiments. Moreover, we repeated each experiment for two significantly different stream arrival rates, yielding a total of four different experiments. The load pattern 1, which is used in experiment 1 and 2, consists of two load peaks, while the load pattern 2, which is used in experiment 3 and 4, has six load peaks. The renting of VMs was based on the hourly charge model of Amazon EC2.

The experiments used both SD (Standard-Definition) and HD video streams. At present, 10% of videos available at YouTube are in HD, while YouTube has more HD content than any other video hosting site [24]. However, in the near future, the ratio of HD versus SD is expected to increase. Therefore, the load generation assumed 30% HD and 70% SD video streams. The video segmentation was performed at the *GOPs* level. The segmentation produced video segments, which were sent to the *transcoding servers* for execution. For HD videos, the average size of a video segment was 75 frames with a standard deviation of 7 frames. Likewise, for SD videos, the average size of a segment was 250 frames with a standard deviation of 20 frames. The total number of frames in a video stream was in the range of 15000 to 18000, which approximates to a video play time of 8 to 12 minutes.

The desired play rate for a video stream is often fixed: 30 frames per seconds (fps) for SD videos and 24 fps for HD videos. Whereas, 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 execution times.

1) Experiment 1: Relatively Normal Load with Low Arrival Rate: The objective of experiment 1 was to simulate a relatively normal load with a low stream arrival rate. It was designed to generate a load representing a maximum of 200 simultaneous video streams in two different load peaks. In the first peak, the streams were ramped-up from 0 to 200, while adding a new stream every 100 seconds. After the ramp-up phase, the number of streams was maintained constant for 1 hour and then ramped-down to 100 streams.

The second peak ramped-up from 100 streams to 200 streams, while adding a new stream every 150 seconds. The ramp-up phase was followed by a similar constant phase as in the first peak. Then, the ramp-down phase removed all streams from the system.

2) Experiment 2: Relatively Normal Load with High Arrival Rate: The objective of experiment 2 was to simulate a relatively normal load, but with a high stream arrival rate. It also generated a load representing a maximum of 200 simultaneous video streams in two different load peaks. However, a new stream was added every 20 seconds in the first peak and every 30 seconds in the second peak.

3) Experiment 3: Highly Variable Load with Low Arrival Rate: Experiment 3 was designed to simulate a load pattern of a highly variable video demand. It generated a load representing a maximum of 290 simultaneous video streams consisting of six different load peaks. In the first peak, the streams were ramped-up from 0 to 170. Then, in the second peak from 110 to 290. Likewise, 210 to 280, 215 to 250, 120 to 200, and 100 to 170, respectively, in the third, fourth, fifth, and sixth peaks. The stream ramp-up rate was 1 new stream per 150 seconds. Each ramp-up phase was followed by a ramp-down phase. Finally, the last ramp-down phase removed all streams from the system.

4) Experiment 4: Highly Variable Load with High Arrival Rate: Experiment 4 used a load pattern similar to that used in experiment 3, except that a new stream was added every 30 seconds.

B. Results and Analysis

In Figures 5, 6, 7, and 8, the number of servers plot shows dynamic resource allocation for the cluster of *transcoding servers*. The transcoding jobs plot represents the total number of jobs in the system at a particular time instance. It includes the jobs in execution on *transcoding servers* and the jobs that are waiting in the queues. The queue waiting time plot shows average queue waiting time of all *transcoding servers*. The target play rate plot shows the sum of target play rates of all video streams in the system. Likewise, the transcoding rate plot represents the total transcoding rate of all servers. The number of completed streams plot depicts how many video streams were successfully played since the start of the simulation, while the number of streams with jitter plot shows the total number of streams that had at least one *transcoding jitter* in them. Similarly, the number of transcoded frames plot represents the total number of video frames that were successfully transcoded since the start of the simulation, while the number of dropped frames plot shows the total number of dropped frames.

1) Experiment 1: Relatively Normal Load with Low Arrival Rate: Figure 5 presents results from experiment 1. It used a maximum of 92 *transcoding servers* for a maximum of 200 simultaneous streams with 0 overloaded servers. There were a maximum of 5052 jobs in the system at a particular time. Moreover, a total of 13140 streams consisting of approximately 1.5×10^6 transcoding operations and 1.8×10^8 video frames were transcoded in 13 hours and 3 minutes of simulated time with 1576 deferred streams and 0 rejected streams. Only 310 streams had jitters in them, which approximates to 2.4% of the total number of streams. The job scheduling algorithm dropped a total of 4×10^7 frames, which constitutes 18% of total frames. The results also show that the actual transcoding rate was always close to the target play rate, which was desirable for our resource allocation algorithm [7]. Therefore, the results indicate that the proposed admission control and job scheduling algorithms prevent overloading of transcoding servers, while at the same time provide a good tradeoff between cost and QoS.

2) Experiment 2: Relatively Normal Load with High Arrival Rate: Figure 6 presents results from experiment 2. It used a maximum of 90 *transcoding servers* for a maximum of 200 simultaneous streams with 0 overloaded servers. There were a maximum of 5119 jobs in the system at a particular time.

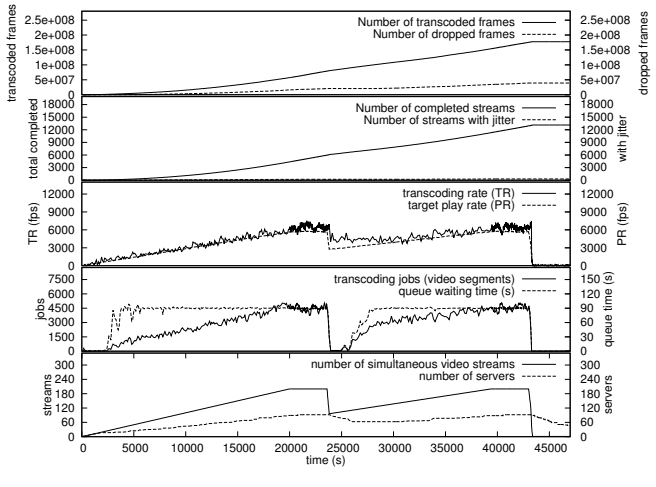

Fig. 5. Experiment 1 results: relatively normal load with low arrival rate

Moreover, a total of 5372 streams consisting of approximately 6×10^5 transcoding operations and 7×10^7 video frames were transcoded in 5 hours and 8 minutes of simulated time with 548 deferred streams and 0 rejected streams. Only 215 streams had jitters in them, which approximates to 4% of the total number of streams. The job scheduling algorithm dropped a total of 1×10^7 frames, which constitutes 15.7% of total frames. Therefore, in this experiment, the proposed admission control and job scheduling algorithms prevented overloading of transcoding servers, while at the same time maintained a good tradeoff between cost and QoS under high arrival rates.

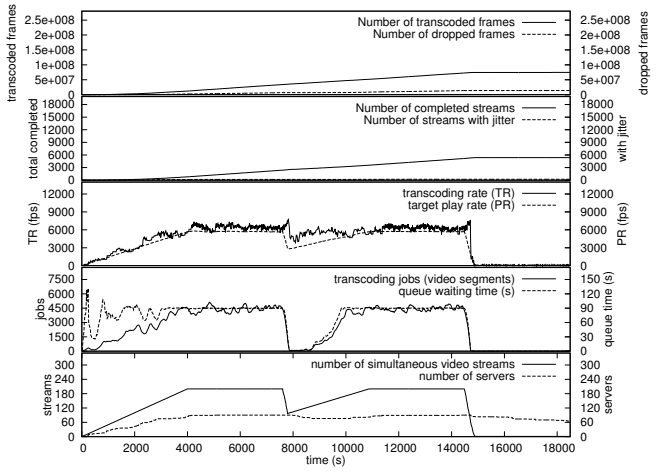

Fig. 6. Experiment 2 results: relatively normal load with high arrival rate

3) Experiment 3: Highly Variable Load with Low Arrival Rate: Figure 7 presents results from experiment 3. It used a maximum of 125 *transcoding servers* for a maximum of 290 simultaneous streams with 0 overloaded servers. There were a maximum of 6792 jobs in the system at a particular time. Moreover, a total of 23620 streams consisting of approximately 2.7×10^6 transcoding operations and 3.3×10^8 video frames were transcoded in 18 hours and 31 minutes of simulated time with 2432 deferred streams and 0 rejected streams. Only 384 streams had jitters in them, which approximates to 1.6% of the total number of streams. The job scheduling algorithm dropped a total of 5.9×10^7 frames, which constitutes 15% of total frames. Therefore, the proposed algorithms prevented overloading of transcoding servers, while at the same time maintained a good tradeoff between cost and QoS under highly variable load.

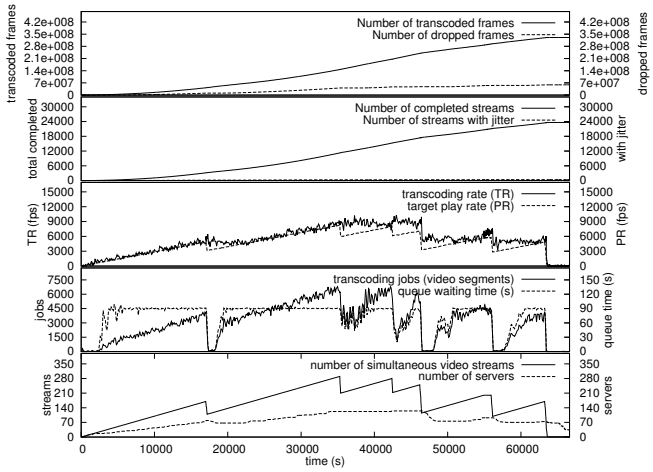

Fig. 7. Experiment 3 results: highly variable load with low arrival rate

4) Experiment 4: Highly Variable Load with High Arrival Rate: Figure 8 presents results from experiment 4. It used a maximum of 119 *transcoding servers* for a maximum of 290 simultaneous streams with 0 overloaded servers. There were a maximum of 6674 jobs in the system at a particular time. Moreover, a total of 5611 streams consisting of approximately 6.3×10^5 transcoding operations and 8.4×10^7 video frames were transcoded in 4 hours and 54 minutes of simulated time with 511 deferred streams and 0 rejected streams. Only 255 streams had jitters in them, which approximates to 4.5% of the total number of streams. The job scheduling algorithm dropped a total of 8.6×10^6 frames, which constitutes only 9.3% of total frames. Therefore, the proposed algorithms prevented overloading of transcoding servers, while at the same time maintained a good tradeoff between cost and QoS under highly variable load with high arrival rates.

VII. Conclusion

In this paper, we presented a stream-based admission control approach and a job scheduling algorithm for video transcoding called SBACS. It uses queue waiting time of individual transcoding servers to make admission control decisions. For preemptive control, SBACS implements a two-step load prediction model, which predicts queue waiting time of individual servers. To reduce over-admission, SBACS implements per

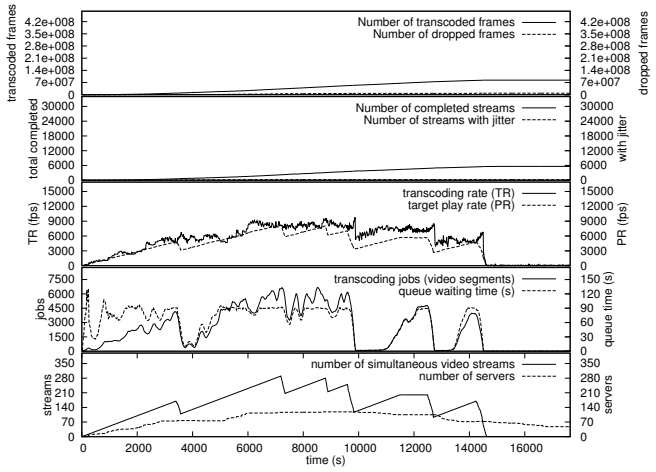

Fig. 8. Experiment 4 results: highly variable load with high arrival rate

stream admission, which makes an admission control decision for each incoming video stream. To ensure efficient utilization of the transcoding servers, video streams are segmented at the *GOP* level. It also provides a stream deferment mechanism, which exploits cloud elasticity to temporarily defer some new streams that would otherwise be rejected. In addition to admission control, SBACS also features a job scheduling algorithm, which complements admission control and prevents *transcoding jitters* in the admitted streams. The algorithm uses temporal resolution reduction transcoding, which drops a small proportion of video frames in a video segment to reduce the required transcoding time. We presented a discrete-event simulation of the proposed approach along with experimental results involving different load patterns and stream arrival rates. The results showed that SBACS provides a good tradeoff between cost and QoS. Moreover, it prevents servers from becoming overloaded, reduces over-admission, reduces rejected streams, and reduces jitters in the admitted streams while dropping only a small proportion of video frames.

ACKNOWLEDGMENT

This work was supported by the Cloud Software Finland research project and by an Amazon Web Services research grant. Adnan Ashraf and Fareed Jokhio were partially supported by the Foundation of Nokia Corporation and by doctoral scholarships from the Higher Education Commission (HEC) of Pakistan.

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] 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.

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

[5] K. Stuhlmuller, N. Farber, M. Link, and B. Girod, "Analysis of video transmission over lossy channels," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 1012–1032, 2000.

[6] "Amazon Elastic Compute Cloud." [Online]. Available: http://aws.amazon.com/ec2/

[7] 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.

[8] 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.

[9] M. Andreolini and S. Casolari, "Load prediction models in web-based systems," in *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, ser. valuetools '06. New York, NY, USA: ACM, 2006.

[10] J. Almeida, V. Almeida, D. Ardagna, I. Cunha, C. Francalanci, and M. Trubian, "Joint admission control and resource allocation in virtualized servers," *J. Parallel Distrib. Comput.*, vol. 70, no. 4, pp. 344–362, Apr. 2010.

[11] T. Voigt and P. Gunningberg, "Adaptive resource-based web server admission control," in *Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on*, 2002, pp. 219 – 224.

[12] A. Robertsson, B. Wittenmark, M. Kihl, and M. Andersson, "Admission control for web server systems - design and experimental evaluation," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 1, dec. 2004, pp. 531 –536 Vol.1.

[13] S. Muppala and X. Zhou, "Coordinated session-based admission control with statistical learning for multi-tier internet applications," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 20 – 29, 2011.

[14] C.-J. Huang, C.-L. Cheng, Y.-T. Chuang, and J.-S. R. Jang, "Admission control schemes for proportional differentiated services enabled internet servers using machine learning techniques," *Expert Systems with Applications*, vol. 31, no. 3, pp. 458 – 471, 2006.

[15] L. Cherkasova and P. Phaal, "Session-based admission control: a mechanism for peak load management of commercial web sites," *Computers, IEEE Transactions on*, vol. 51, no. 6, pp. 669 –685, jun 2002.

[16] X. Chen, H. Chen, and P. Mohapatra, "ACES: An efficient admission control scheme for QoS-aware web servers," *Computer Communications*, vol. 26, no. 14, pp. 1581 – 1593, 2003.

[17] Y. A. Shaaban and J. Hillston, "Cost-based admission control for internet commerce QoS enhancement," *Electronic Commerce Research and Applications*, vol. 8, no. 3, pp. 142 – 159, 2009.

[18] 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.

[19] 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.

[20] 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.

[21] D. Montgomery, E. Peck, and G. Vining, *Introduction to Linear Regression Analysis*, ser. Wiley Series in Probability and Statistics. John Wiley & Sons, 2012.

[22] 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.

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

[24] "35 mind numbing youtube facts, figures and statistics infographic," 2012/05/23. [Online]. Available: http://www.jeffbullas.com/2012/05/23/35-mind-numbing-youtube-facts-figures-and-statistics-infographic/