Joint Online Transcoding and Geo-distributed Delivery for Dynamic Adaptive Streaming

Zhi Wang, Lifeng Sun, Chuan Wu*, Wenwu Zhu, and Shiqiang Yang
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University
*Department of Computer Science, The University of Hong Kong

***Abstract*—Dynamic adaptive video streaming has emerged as a popular approach for video streaming in today's Internet. To date the two important components in dynamic adaptive streaming, video transcoding which generates the adaptive bitrates of a video and video delivery which streams the videos to users, have been separately studied, resulting in a huge waste of computation and storage resource due to transcoding useless videos and suboptimal streaming quality due to homogeneous video replication. In this paper, we propose to jointly perform video transcoding and video delivery for adaptive streaming in an online manner. We conduct extensive measurement studies of a video sharing system and a CDN to motivate our design. We formulate and solve optimization problems to enable high streaming quality for the users, and low computation and replication costs for the system. In particular, our design connects video transcoding and video delivery based on users' preferences of CDN regions and regional preferences of video versions. Extensive trace-driven experiments further confirm the superiority of our design.**

I. INTRODUCTION

Dynamic adaptive streaming over HTTP (DASH) has emerged as a popular video streaming method [1], in which content providers can leverage the largely-deployed CDN servers to cache and deliver the video segments [2]. Adaptive streaming has been widely implemented and supported by the industry, including Apple HTTP Live Streaming, Microsoft Live Smooth Streaming, and Adobe Adaptive Streaming. It allows users with heterogeneous and dynamically changing network conditions to receive an adaptive bitrate, achieving the best video streaming experience in different contexts [3].

In adaptive streaming, video service providers have to not only deliver the video *segments* (data blocks in a video that can be downloaded over HTTP and played independently), but also transcode the videos to different *versions* (*i.e.*, videos with different bitrates of the same content) for the users. In this paper, we refer to *transcoding* as transcoding a video to different bitrates, which may consume a lot of computation resource [4]. To date, traditional approaches separately perform the video transcoding and delivery — being unaware of which segments users will request, they have to transcode every video published to a set of fixed versions, and replicate segments of different versions using the same strategy.

Problems of the traditional approaches for adaptive streaming are as follows: (1) Prefixed versions only allow users to choose from a small set of candidate bitrates, which cannot effectively adapt to the changing network conditions. (2) To address this problem, video providers increase the number of adaptive versions — as both the number of uploaded videos and the number of versions are increasing, a huge amount of computation resource is required to transcode all the videos to all the versions [5]. As the popularity distribution of video segments is becoming significantly heavy-tailed, *i.e.*, a substantial fraction of video segments are not requested at all — pre-transcoding them could be a huge waste of valuable computation resource. The situation is exacerbated by today's user-generated-content (UGC)-based video sharing services [6]. (3) On the other hand, traditional approaches are not aware of the users' *preferences* of different *peering servers* (*i.e.*, servers directly uploading the video segments to users) when users receive segments of different versions, leading to the mismatch between the download speed and the required segment bitrate, *e.g.*, a user being able to receive a high-bitrate segment might be redirected to a peering server with a slow connection to the user [7].

To address these problems, we propose to jointly schedule segment transcoding and delivery in an online manner, using geo-distributed computation and communication resources. The new design philosophy allows us to jointly optimize the streaming quality for users and minimize the computation and bandwidth cost for transcoding and replicating the video segments. To study the benefits of our proposal for real-world video providers, we measure the user request patterns of adaptive video streams in a representative video streaming service in China, BesTV [8] (an IPTV system serving over 16 million users). We have made the following observations: (**i**) due to the skewness of popularity distribution of the videos, segments and versions, the online transcoding paradigm has the potential to significantly reduce the demand for computation resource.

To demonstrate that our proposal can be implemented

Zhi Wang has graduated and is now with Graduate School of Shenzhen, Tsinghua University. We thank BesTV and Tencent for providing the traces used in our study. We acknowledge Alan Zhuang from Tencent for the insightful discussions. This work is supported in part by the National Basic Research Program of China (973) under Grant No. 2011CB302206, the National Natural Science Foundation of China under Grant No. 61133008, 61210008, and 61272231, "1000 People Plan" grant under Grant No. 553360001, the research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, Beijing Key Laboratory of Networked Multimedia, and a grant from Hong Kong RGC under the contract HKU 717812E.

practically in today's CDNs which are already widely used for adaptive video streaming, we further measure the availability of computation and bandwidth resources in Tencent CDN [9], which serves over 70% of the traffic from one of the largest content providers in China. We have further made the following observations: (**i**) A substantial amount of idle computation resource can be provided by the *backend servers* (*i.e.*, servers supporting the peering servers), and the idle computation resource is relatively stable over time, indicating that online transcoding can be effectively performed by these backend servers; (**ii**) Since peering servers are deployed at different geographic locations with different Internet Service Providers (ISPs), and scheduled to serve different numbers of user requests, the distribution of users' download speeds differs across different CDN regions, *i.e.*, (1) users have different preferences of CDN servers in different regions from which to receive segments, while (2) servers in different regions have different preferences of versions of video segments to transcode.

To the best of our knowledge, our study is the first to explore the new design space of joint online transcoding and geo-distributed delivery. Our contributions are summarized as follows: (1) We conduct large-scale measurement studies to motivate our approach, the feasibility of its implementation, and the guidelines for our design; (2) To achieve good streaming qualities, low computation resource consumption, and low video segment replication cost, we connect video transcoding and video delivery based on users' region preferences and regional version preferences — we use users' preferences of regions to redirect them to their ideal peering servers, and we use the regional version preferences to schedule the transcoding tasks; (3) We formulate the problems, design practical algorithms to solve them, and demonstrate the performance of our design.

The rest of the paper is organized as follows. We present the measurement insights that motivate our design in Sec. II. We present our detailed design in Sec. III. We verify the effectiveness and evaluate its performance in Sec. IV. We discuss related works in Sec. V. Finally, we conclude the paper in Sec. VI.

II. Measurements and Observations

We conduct measurement studies to motivate our design, and summarize design principles learnt.

A. Measurement Setup

To demonstrate the benefits and feasibility of our proposal, we use large-scale measurement studies based on valuable traces collected from BesTV and Tencent CDN.

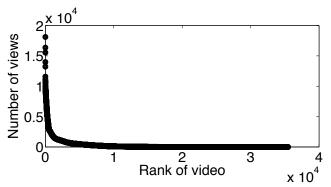
1) Traces of Users' Video Viewing Patterns: To study the potential of using an online transcoding scheme to save computation resource, we have collected real-world traces on video access patterns in BesTV. In BesTV, videos are published into 17 categories, and pre-transcoded into 4 versions (including 700 Kbps, 1300 Kbps, 2300 Kbps and 4000 Kbps). We collected viewing activities of users in Heilongjiang province in November 2012, about how 190K videos were watched by users from over 3 million IP addresses. For each of the streaming sessions, the traces record which segments were downloaded by which users, including the time stamp when a segment was downloaded, the user ID, the video ID, the size and version of the segment, and the time spent on downloading the segment. Using these traces, we can show the great potential of our joint transcoding and streaming paradigm in Sec. II-B1.

2) Traces of CDN Characteristics: To study the feasibility of online transcoding in a CDN system, which has already been widely used for adaptive streaming [2], we collected traces of the backend and peering servers from Tencent CDN, as follows: (1) *CPU load patterns*. To study the computation resource availability for segment transcoding, we collected the CPU load traces from the backend servers in Tencent CDN. In particular, the CPU load of $5,441$ servers was recorded every 5 minutes, for the whole month of March 2013. Each CPU load trace item contains the following information: timestamp and the CPU load recorded as the average number of processes waiting on each CPU core, *e.g.*, a CPU load greater than 1 indicates that the server is fully loaded. (2) *Bandwidth patterns*. To study the users' preferences of CDN regions, and the regional preferences of versions to transcode, we have collected traces including 3.39 billion TCP connections from peering servers located at 55 regions in May 2013. These TCP connections were established to download contents with sizes varying from tens of bytes to 4.8 GB. Each of the trace items contains the following information: the timestamp indicating when a TCP connection was established, the client IP, the number of downloaded bytes and the connection duration. In Sec. II-B2, we use these traces to study the feasibility and provide guidelines for our design.

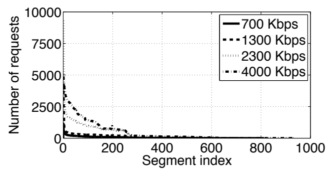
B. Measurement Insights

1) Potential – Computation Resource to Be Saved: Based on the video viewing records in BesTV, Fig. 1(a) illustrates the popularity distribution of the videos. Each sample represents the number of user requests of a video in one month versus the rank of the video. We observe that over 53% of these videos had no viewer in a month. This can be explained by the fact of today's video sharing services that the time users spent on watching videos grows much slower than the growth of the number of videos, and such skewness of the popularity distribution is also prevalent in other UGC-based video sharing systems, such as YouTube [6].

In Fig. 1(a), only 13% of the videos have a monthly view number larger than 500. We further investigate how different segments with different versions inside such a relatively popular video (with about $1,000$ segments), are requested by users. Fig. 1(b) illustrates the distribution of the requests for segments of one of the most popular videos. Each curve represents the number of segment requests versus the segment index. We observe that (1) only a small range of segments are requested by many users, *e.g.*, the first tens of the segments; (2) different versions receive different numbers of requests,

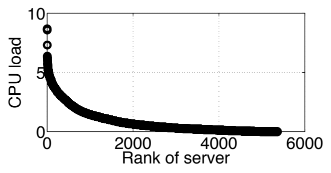


(a) Number of user views versus the rank of video.

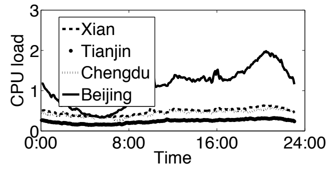


(b) Number of segment requests versus segment index for different versions.

Fig. 1. Popularity of videos, segments and versions in BesTV (Heilongjiang, November 2012).

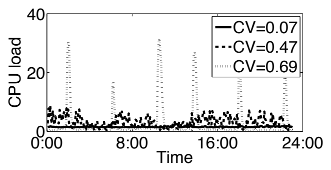


(a) Average CPU load in 15 minutes versus the rank of the server (8PM, March 5, 2013).

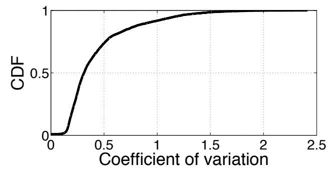


(b) Average CPU load on servers of different regions over time (March 5, 2013).

Fig. 2. Average CPU load of backend servers.



(a) Examples of server CPU load over time.



(b) The CDF of the coefficient of variation of the server CPU load.

Fig. 3. Variation of server CPU load over time.

e.g., the version of 4000 Kbps is requested by much more users; and (3) a large fraction of segments are requested by nobody for some versions, *e.g.*, the last segments of the 700 Kbps and 1300 Kbps versions.

These observations indicate that as more and more videos are published, by both the professional content providers and individuals, pre-transcoding every segment of all videos into an increasing number of versions can be a huge waste of computation resource, motivating our segment-based online transcoding.

2) Possibility – Computation Resource Available in CDN: Inspired by Akamai's realtime monitoring [10] that the CPU load of CDN servers not only can be efficiently measured but also is diverse across different servers, we explore to perform video transcoding using idle computation resource on the CDN servers.

First, we demonstrate the availability of idle computation resource in the CDN. Fig. 2(a) plots the CPU load — the average number of processes waiting on each CPU core of the backend servers in a time slot of 15 minutes. We observe that in a particular time slot, the CPU load of the $5,441$ backend servers is varying from around 0 to 8.6, and as many as 72.4% (*resp.* 55.9%) of backend servers have a CPU load smaller than 1.0 (*resp.* 0.5), indicating that a substantial amount of available computation resource in the CDN can be allocated for video transcoding. The reason for the high availability of the computation resource in a CDN is that many backend servers are only assigned with simple I/O tasks, *e.g.*, loading data from the distributed storage system for the peering servers.

We further study the availability of computation resource in a region of the CDN. We use a city-ISP pair to identify a region. Fig. 2(b) illustrates the regional CPU load, *i.e.*, the average CPU load of all the backend servers in that region. In this figure, each curve represents the CPU load of the four largest regions, *i.e.*, Xian, Tianjin, Chengdu and Beijing, in one day. We also observe the existence of available computation resource at the region level; meanwhile, we observe that the CPU load differs across different regions, *e.g.*, the CPU load of Xian is much lower than that of Beijing on that day.

To make use of the idle CPU resource from the backend servers in the CDN for segment transcoding, we also need to investigate the stability of the idle computation resource on the backend servers. Since these servers can be scheduled to run different tasks, the available computation resource provided by these servers may vary over time. We use an average coefficient of variation to evaluate the daily churning level of the CPU load of a backend server, calculated as follows: $CV = 1/24\sum_{h=0}^{23}\sqrt{E[(X_h - \bar{X_h})^2]}/\bar{X_h}$, where X_h represents a set of CPU load records of a particular server in one hour h, *i.e.*, there are 12 samples in an hour, as CPU load is recorded every 5 minutes. A large CV implies a highly churning CPU load over time. In Fig. 3(a), we sample 3 servers with different CV's on March 5, 2013. Each curve represents the CPU load of the server over time. We observe that servers with $CV = 0.07$ and $CV = 0.47$ have a stable CPU load over time; while the server with $CV = 0.69$ has a relatively churning CPU load, varying from 0.17 to 31.5 in several minutes.

We investigate the distribution of CVs of all the backend servers in the CDN. In Fig. 3(b), the curve represents the CDF of CVs of all the servers on the same day. We observe that over 70% of the servers have a CV smaller than 0.5, indicating that the CPU load of many backend servers is relatively stable — their capacity for performing video transcoding in the near future can be predicted. We use such information in our design of transcoding task scheduling.

3) Connections – Users' Region Preferences and Regional Version Preferences: In the context of online transcoding, we are allowed the degree of freedom that segments can be transcoded by different CDN regions. Next, we explore the guidelines for such transcoding schedule.

$\triangleright$ *Users' preferences of different CDN regions*. Which version of a video segment is requested by a user depends on the user's download speed. Based on the TCP traces of the peering servers, we compare the download speed of about 150 users who downloaded from different peering servers in the same 10 minutes on May 4, 2013. In Fig. 4, each sample is the average download speed of a user downloading from a peering server deployed in Shanghai, versus the average download speed of the same user downloading from a Shenzhen peering server, both with the same ISP. We observe that for over 79% of the users, their download speeds differ over 2 times when they download from servers located at different regions,

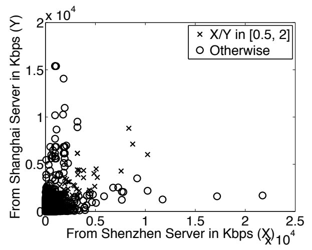


Fig. 4. Comparison of average download speed of users downloading from different peering servers in the same 10 minutes on May 4, 2013.

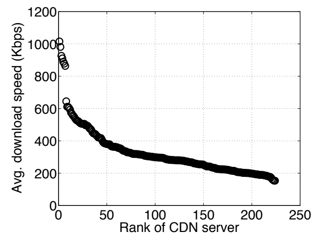


Fig. 5. Average user download speed of different peering servers on May 4, 2013.

indicating that redirecting users to their ideal peering servers can help users receive a better streaming quality.

▷ *Regional preferences of different versions to transcode.* On the other hand, to study the regional preference of versions to transcode, we calculate the average download speeds of users downloading from the peering servers. In Fig. 5, each sample represents the average download speed in one day, of all users served by a server versus the rank of the server. We observe that the average download speed varies quite significantly across these peering servers, from 170 Kbps to 1.1 Mbps.

We further study the download speeds from servers in different regions, *i.e.*, the download speeds from servers in different regions to users. In Fig. 6, each bar represents the minimal, average and maximal download speeds from the peering servers in a region. We observe that the average download speeds across different regions vary from 180 Kbps (region BJT, *i.e.*, Beijing, China Telecom) to 512 Kbps (region ZJM, *i.e.*, Zhejiang, China Mobile). Different regions "cover" users with quite different speeds, *e.g.*, BJT serves most of the low-bitrate users while ZJM serves hit-bitrate users. The rationale behind these observations is as follows: (1) Peering servers are physically deployed at different locations and with different ISPs, so that the Internet connectivity and average bandwidth capacity are different; and (2) Peering servers at different regions are generally scheduled to serve different numbers of user requests, leading to the different server load.

These observations indicate that servers in different regions have different preferences of versions to transcode, *e.g.*, a region with a low CDN-to-user download speed may prefer to produce low-bitrate segments. Satisfying such preferences of regions can reduce the *cost* of replicating transcoded segments, since segments are already transcoded where they are to be requested.

III. Joint Online Transcoding and Geo-Distributed Delivery

A. Framework

Fig. 7 illustrates our design, where segments in different versions of videos are transcoded upon users' requests. In this example, $s1, s2, s3, s4$ represent segments of different versions, which are requested by a user during her streaming session. $R1$, $R2$ and $R3$ are CDN *regions* (each is represented

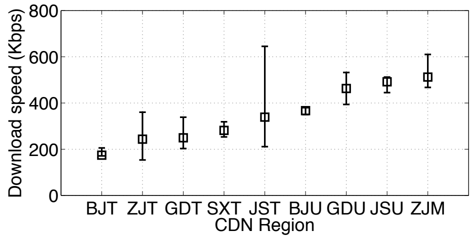


Fig. 6. Average user download speed in different CDN regions on May 4, 2013.

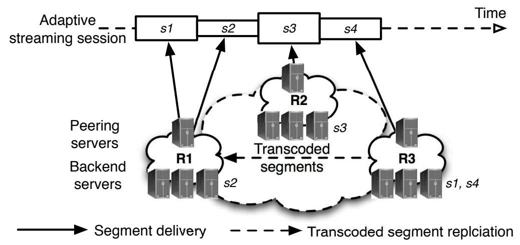


Fig. 7. Joint online transcoding and geo-distributed streaming: an illustration

by a pair of a geographical location and ISP) where servers are deployed with backend servers and peering servers deployed. Segments can be transcoded by selected regions (*e.g.*, $s2$ is transcoded by region $R1$), replicated between regions (*e.g.*, $s1$ is replicated from region $R3$ to $R1$), and delivered to users, all in an online manner.

Fig. 8 further illustrates the framework of our online transcoding and delivery scheme, which schedules segment transcoding and replication periodically: based on the recent information collected in time slot $T-1$, we perform transcoding and replication of segments that are likely to be requested in time slot T. The collected information includes: (1) Users' preferences of different regions to receive segments. In our design, we allow users to use a bandwidth estimation approach (*e.g.*, abget [11] which uses little bandwidth to measure the end-to-end bandwidth) to rank a set of candidate peering servers, in the descending order of the estimated download speed. (2) The number of requests for a particular segment, which can be predicted according to users' segment requests in previous time slots. Based on the estimation of CDN-to-user bandwidth and users' segment requests, we are able to estimate which segments will be requested by users in the next time slot, and the request level of each segment. (3) The idle computation resource. As many backend servers have stable CPU load over time according to our measurement study, we use the level of computation resource in the current time slot as the available computation resource in the next time slot. Other regression models (*e.g.*, ARIMA [12]) can also be explored to achieve better prediction accuracy, which we will investigate in the future.

Using such information, we perform the following: (1) *User redirection.* To enable high-quality streaming, our design redirects users to their ideal regions so that they can receive segments at high bitrates. We redirect users at a region level, *i.e.*, a region with the highest CDN-to-user bandwidth will be selected serve a user's request, and peering servers in the

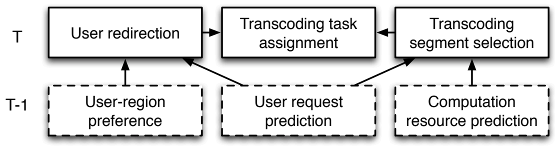

T
T-1
User redirection
Transcoding task assignment
Transcoding segment selection
User-region preference
User request prediction
Computation resource prediction

Fig. 8. Framework of our joint online transcoding and delivery mechanism.

same region are assigned to serve user requests in a round-robin manner. (2) *Transcoding segment selection*. Backend servers with idle CPU resource performs video transcoding, by slicing a video into multiple closed groups of pictures (GoPs), each of which can be transcoded independently [13]. To allow a smooth playback, when a segment request of a particular version is not transcoded timely, we send the user the segment of an alternative version, whose bitrate is closest to that of the requested version. We prioritize more "important" segments that are more critical to users' streaming quality, to be transcoded when computation resource is not sufficient. (3) *Transcoding task assignment*. A transcoded segment is cached by the backend servers and replicated to other regions, according to our replication strategy. Transcoding is performed by strategically selected regions, so that the cost of replicating the transcoded segments to other regions can be minimized.

Before we present the design details, Table I gives important notation used in this paper.

TABLE I
IMPORTANT NOTATIONS.

Symbol	Definition
$\mathbf{U}^{(T)}$	Set of users requesting segments in time slot T
$D^{(T)}(u,r)$	Binary variable indicating whether user u will download from region r in time slot T
$H(u,r)$	Preference level for user u to receive video stream from region r
W_r	Bandwidth capacity of region r
$e^{(T)}_{(s,v)}$	Importance level of a particular segment (s,v) in time slot T
$Q^{(T)}_{(s,v)}$	Number of requests of segment (s,v) from all regions in time slot T
$Y^{(T)}_{(s,v)}$	Quality gain if segment (s,v) is transcoded in time slot T
$B(v)$	Bitrate of a particular version v
$\mathbf{G}^{(T)}(s)$	The set of transcoded versions of segment s
$\mathbf{R}$	Set of CDN regions
$\mathbf{E}^{(T)}$	Set of segments to be transcoded in time slot T
$L(u,r)$	Highest version that u can receive when she downloads from region r
$C(s,v)$	Computation resource required to perform the transcoding task to generate a segment s of version v
$I^{(T)}(r)$	Available computation resource that can be allocated for video transcoding from region r in time slot T
$F[(s,v),r]$	Overall replication cost when segment (s,v) is transcoded in region r
$A^{(T)}_{(s,v)}$	Region assigned to transcode segment (s,v) in time slot T

B. Quality-Driven Redirection

In our design, taking the advantage of online transcoding, a user can be redirected to her ideal region where segments are generated on the fly. This design principle allows users to choose a CDN region with the largest download speed to receive the segments, without considering the segment availability.

We first formulate this problem in a centralized manner. We denote $\mathbf{U}^{(T)}$ as the predicted set of users requesting different segments in the system in time slot T, and $\mathbf{R}$ as the set of CDN regions where a user can be redirected. We use $D^{(T)}$ to denote a redirection strategy, where the binary variable $D^{(T)}(u,r) = 1$ (*resp*. 0) indicates that user u will (*resp*. will not) be downloading from region r in the next time slot T.

In the context of adaptive video streaming, we assume that users expect to receive a large bitrate for good streaming quality whenever possible. Thus, we use $H(u,r)$ to denote user u's preference to download from a CDN region r. $H(u,r)$ can be defined as a concave increasing function of the estimated download speed achieved when user u downloads from peering servers in region r. We formulate the region-level user redirection as an optimization problem, as follows:

$$\max_{D^{(T)}} \sum_{u\in\mathbf{U}^{(T)},r\in\mathbf{R}} H(u,r)D^{(T)}(u,r), \tag{1}$$

subject to:

$$\sum_{r\in\mathbf{R}} D^{(T)}(u,r) \le 1, \forall u \in \mathbf{U}^{(T)},$$

$$\sum_{u\in\mathbf{U}^{(T)}} D^{(T)}(u,r)B(L(u,r)) \le W_r, \forall r \in \mathbf{R},$$

where $B(v)$ is the bitrate of version v, $L(u,r)$ is the version with the highest bitrate that u can receive when she downloads from region r, and W_r is the bandwidth capacity of CDN region r. The rationale of the optimization is to maximize the streaming quality for users by the redirection.

This problem is generally NP-hard, since we can reduce a conventional 0-1 knapsack problem which is NP-hard to it. We design an algorithm to heuristically solve it in a distributed manner: (1) When a user starts to watch a video, the system assigns her a list of candidate peering servers from regions with the lowest load. (2) The user ranks these servers in descending order of the estimated download speeds as discussed in Sec. III-A, and sends connection requests to these servers. (3) On the other hand, a peering server may receive connection requests from many users, and can only accept a limited number of users according to its available bandwidth W_r. User u is prioritized to be accepted if she has a larger $H(u,r)/B(L(u,r))$ with the CDN region r — this value reflects a marginal "gain" in streaming quality by a unit of bandwidth allocated. (4) The user selects the best peering server from the ones accepting her request according to the ranked list. In a real system, this algorithm can be effectively implemented and executed in a distributed manner.

C. Region-Preference-Aware Transcoding Schedule

After users are redirected to the CDN regions, they send requests for video segments of different versions. Based on

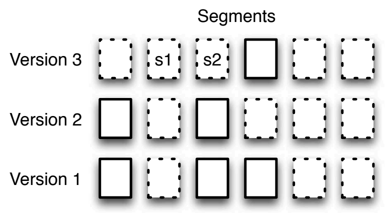


Fig. 9. Segment importance in the context of online transcoding.

the segment request prediction, we perform the transcoding task schedule, which works in two steps: (1) we prioritize the segment transcoding tasks such that important segments are transcoded more urgently; and (2) we distribute the transcoding tasks to CDN regions, such that segments are transcoded where they are more likely to be requested.

1) Prioritizing Segment Transcoding Tasks: We prioritize the segment transcoding tasks according to the importance of these segments. We denote $e^{(T)}_{(s,v)}$ as the importance level of segment s of version v in time slot T. $e^{(T)}_{(s,v)}$ depends on the following factors: (1) the estimated number of user requests for the segment, discussed in Sec. III-A; and (2) the quality-wise importance of the segment, which depends on the existing versions of the same segment. In particular, $e^{(T)}_{(s,v)}$ can be calculated as follows:

$$e^{(T)}_{(s,v)} = Q^{(T)}_{(s,v)} Y^{(T)}_{(s,v)},$$

where $Q^{(T)}_{(s,v)}$ denotes the predicted number of requests of the particular segment (s, v) in the next time slot T, and $Y^{(T)}_{(s,v)}$ is the quality gain if the segment is transcoded to version v. Fig. 9 illustrates an example of the importance of a segment: a solid block represents a segment transcoded, and a dashed block represents one that is not transcoded yet. When segment $(s1, v3)$ and $(s2, v3)$ are both requested by the same number of users, $(s1, v3)$ is prioritized to be transcoded over $(s2, v3)$, since users requesting $(s2, v3)$ can be served by an alternative version $(s2, v2)$, which has a close bitrate to the original requested one, and no version of segment $s1$ exists in the system.

In our design, $Y^{(T)}_{(s,v)}$ is calculated as the "mismatch" level of the bitrate if v is not transcoded as follows:

$$Y^{(T)}_{s,v} = \begin{cases} \min_w (B(v) - B(w))/B(v), & \exists w \in \mathbf{G}^{(T)}(s), w < v \\ \gamma, & \text{otherwise} \end{cases},$$

where $\mathbf{G}^{(T)}(s)$ is the set of all the versions of the segment existing in the system. When there is a lower-version replacement, a large $Y^{(T)}_{(s,v)}$ indicates that users will receive a highly mismatched bitrate if the version v is not transcoded, such that version v is important to segment s quality-wise. When there is no lower-version replacement, $Y^{(T)}_{(s,v)}$ is assigned with a large value γ, indicating that no replacement version of the segment has been transcoded.

Based on the definition of the importance level of segments, we determine which segments to be transcoded by formulating it as an optimization problem, as follows:

$$\max_{\mathbf{E}^{(T)}} \sum_{(s,v)\in\mathbf{E}^{(T)}} e^{(T)}_{(s,v)}, \tag{2}$$

subject to:

$$\sum_{(s,v)\in\mathbf{E}^{(T)}} C(s,v) \le \sum_r I^{(T)}(r),$$

where $\mathbf{E}^{(T)}$ is the set of segments to be transcoded in time slot T, $C(s, v)$ is the amount of computation resource required to transcode a segment (s, v), and $I^{(T)}(r)$ is the aggregated idle computation resource from the CDN region r. According to [13], it takes different CPU times to generate different segments in the same video. In our design, we use the average CPU time spent on generating historical segments of a particular version and size, to estimate the computation resource required to transcode any segment with that version and size.

The rationale of the optimization that is also a 0-1 knapsack problem is to select a set of segments that are the most important ones in the next time slot T. We design the following algorithm to solve this problem: (1) We collect the information for prediction in a centralized manner, *e.g.*, users (*resp.* backend servers) report which segments they are downloading (*resp.* the CPU load information) to a centralized server, which will carry out the prediction; (2) Based on the prediction, we rank the requested segments in descending order of $e^{(T)}_{(s,v)}/C(s, v)$; (3) We iteratively select segments from the ranked list to transcode, and update computation resource consumption, until the available idle computation resource is used up.

2) Scheduling Transcoding Tasks across Regions: After the tasks are selected, they are to be scheduled to different regions where backend servers can provide the computation resource. Without lose of generality, we use $A^{(T)}_{(s,v)}$ to denote the region where segment (s, v) will be transcoded — the segment will be replicated from this region which originally stores the transcoded version, to other regions where users request it .

According to our measurement studies in Sec. II, heterogeneous preferences of video versions exist at different regions, due to the different download speeds from the servers at different regions. As a result, it is promising to strategically assign transcoding tasks for different segments to backend servers at different CDN regions for a minimized replication cost.

We use $F[(s, v), r]$ to denote the overall replication cost if segment (s, v) is transcoded in region r. It can be calculated as follows:

$$F[(s,v),r] = \sum_{r'\neq r, \mathbf{J}^{(T)}_{(s,v),r'} > \beta} Z_{r,r'}(s,v),$$

where $\mathbf{J}^{(T)}_{(s,v),r'}$ is the number of requests of segment (s, v) to be served by a region r', $Z_{r,r'}(s, v)$ represents the replication cost when segment (s, v) is replicated from region r to region r', depending on the size of the segment and the bandwidth between CDN regions r and r' [14]. β is a threshold of the number of requesting users from a region to trigger a replication. $\mathbf{J}^{(T)}_{(s,v),r'}$ can be derived from the optimization in (1), which determines the redirection of users. The rationale of this definition is that, in our design, a transcoded segment can be replicated from where it is transcoded to other regions where it is substantially requested (*i.e.*, $\mathbf{J}^{(T)}_{(s,v),r'} > \beta$) — a large $F[(s,v),r]$ indicates a large replication cost between CDN regions if segment (s,v) is transcoded by region r. The task assignment problem is then formulated as follows:

$$\min_{A^{(T)}} \sum_{(s,v)\in\mathbf{E}^{(T)}} F[(s,v), A^{(T)}_{(s,v)}], \tag{3}$$

subject to:

$$\sum_{(s,v)\in\mathbf{E}^{(T)}} C(s,v) \le I^{(T)}(r), \forall r \in \mathbf{R}.$$

The rationale of the optimization is to schedule the segment transcoding tasks to different CDN regions, so that the overall replication cost can be minimized. In our implementation, we also design a practical algorithm to heuristically solve the problem, as follows: (1) We first rank all the pairs of the CDN regions and segments (*i.e.*, $|\mathbf{R}|\left|\mathbf{E}^{(T)}\right|$ elements), in ascending order of $F[(s,v),r]$; (2) we pick the region-segment pair "$r-(s,v)$" with the smallest $F[(s,v),r]$ and assign the transcoding task of segment (s,v) to region r; (3) we update the available computation resource of the selected region, and iteratively perform (2) until all computation resource in all the regions is fully used up. This algorithm can be implemented in a centralized manner, where a central server is deployed to collect the request information from streaming servers and make the decisions. Such implementation has been well applied in peer-assisted on-demand streaming systems [15], where a central server tracks the storage status of peers to help them find each other.

In our design, regions with a request number of a segment larger than β will serve a replication of the segment; while for other regions with numbers of requests smaller than β, they will further redirect the users to other regions with the segment transcoded or replicated, according to the users' preferences.

IV. Performance Evaluation

A. Experiment Setup

We develop an event-driven simulation platform which takes users' viewing activities, and the transcoding and redirection decisions as events to drive the experiments. We compare our design with a pre-transcoding baseline scheme. Details are as follows.

$\triangleright$ *Users*. According to models summarized from user viewing traces in BesTV, we simulate $10,000$ users, each of whom repeatedly joins different video sessions. After a user joins the system, she selects a video to watch according to the video popularity distribution. The indices of the first segments users start to view also follow a zipf distribution, with a shape parameter 1.29. When playing a video, a user plays (downloads) sequentially the segments, and may jump to a rand segment ahead with a probability of 0.05. The rationale is that in a video session, how users request segments follow a pattern that users generally play forward and issue a few seeks, most of which are forward seeks [16]. Before leaving a video session, the number of segments a user downloads follows a zipf distribution with a shape parameter 1.12.

$\triangleright$ *Video Provider*. New videos are published every $10,000$ time slots. The popularity of videos follows a zipf distribution with a shape parameter 1.76. In our experiments, the default number of segments in each video is 200, and the default number of versions is 4, if not specified otherwise. The bitrates of the versions are uniformly distributed between the lowest user download speed, and the highest user download speed. The segment length is 10 seconds, and the computation resource required to transcode a segment is randomly distributed within $[5,10]$ CPU seconds [17].

$\triangleright$ *CDN Regions*. We simulate 30 regions. We set a region-to-user average download speed according to the download speed of $10,000$ IP prefixes randomly selected from the CDN traces, *i.e.*, the download speed of an IP prefix is the average download speed of users with the same prefix in a one-week time span, varying from 70 Kbps to 2.2 Mbps. In our experiments, the aggregated CDN bandwidth is sufficient for all the users to stream at their ideal bitrates, and we randomly divide the bandwidth allocation across the regions. We assign the replication cost between each pair of regions within $[0,1]$, and a replication parameter $\beta = 10$. A region has a varying idle computation resource over time with CV randomly selected in $[0,0.5]$, and the average amount of computation resource will be presented in the experiments.

Baseline Algorithm. We compare our design with a general pre-transcoding and load-based redirection strategy: (1) For segment transcoding, all versions of the videos are transcoded before publication, and each transcoded segment is replicated to 3 initial regions randomly selected (*i.e.*, the pre-transcoding scheme); (2) For user redirection, when requesting a segment, a user is redirected to a region which currently has the highest available upload bandwidth (*i.e.*, the load-based redirection scheme).

B. Experiment Results

1) Saving of Computation Resource: In this experiment, we assume that the CDN can provide unlimited computation resource when transcoding is performed, such that we can satisfy all the segment requests of users. In Fig. 10, the curves represent computation resource saved by our design under different number of versions, compared with the pre-transcoding scheme. In particular, each sample is the fraction of computation resource that has been saved over the computation resource required to transcode videos to all the versions till a simulation round. We observe that as the number of versions increases, the computation resource saved by online transcoding increases, *e.g.*, over 90% of the computation resource can be saved when the number of versions is over 8. The reason is that transcoding segments with no viewer to

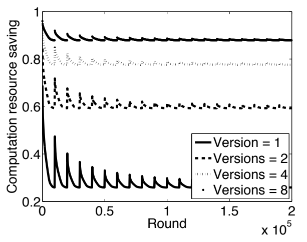


Fig. 10. Computation resource saved by online transcoding under different number of versions.

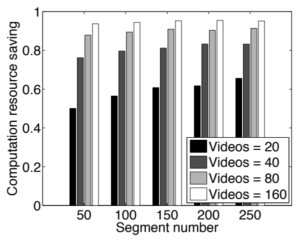


Fig. 11. Computation resource saved by online transcoding under different numbers of videos and segments.

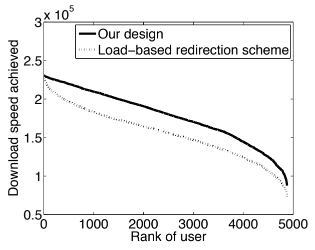


Fig. 12. Comparison of download speed achieved at users under different redirection strategies.

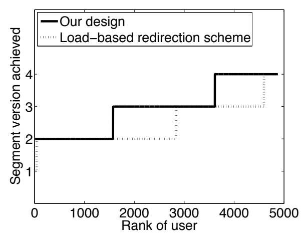


Fig. 13. Comparison of best versions achieved at users under different redirection strategies.

many versions costs a large amount of computation resource. We also observe that the amount of computation resource saving decreases in the first several rounds when new videos are published, and becomes stable afterwards. The reason is that in our design, computation resource is mainly used to transcode the most popular segments after the videos are published, and users who watch videos later largely request the segments that have already been transcoded.

Then, we investigate the impact of the number of videos published each time and the number of segments in each video. In this experiment, we fix the number of versions to 4. In Fig. 11, each bar represents the computation resource saving when a particular number of videos are published each time. We observe that publishing a large number of videos per time slot generally leads to larger computation resource saving. The reason is that the popularity distribution of the videos is heavy-tailed, and more videos with no viewer cause more waste of computation resource with the pre-transcoding scheme.

2) Streaming Bitrates at Users: Taking advantage of online transcoding, users are redirected to their ideal regions to download the videos. We compare our redirection strategy with the load-based redirection scheme. In Fig. 12, each curve plots the average download speed achieved at users versus the rank of users. We observe that our strategy can effectively schedule users to their ideal regions, with an average 181 Kbps improvement of download bandwidth than the load-based redirection scheme. The reason is that the load-based redirection scheme only considers segment replication and available bandwidth of the regions, while our strategy allows users to choose their ideal regions.

Furthermore, we compare the best versions users receive under different redirection strategies. Again, we fix the number of versions to 4. As illustrated in Fig. 13, each curve represents the version downloaded versus the user rank. We observe that as many as 44.8% of the users receive a version of a higher bitrate with our strategy than that with the load-based redirection scheme. In particular, over 4.5x users receive the version with the highest bitrate with our redirection strategy than with the load-based redirection scheme.

3) Fitness of the Transcoded Segments: In the following experiment, we will evaluate the effectiveness of our transcoding task schedule, in how well the transcoded segments match the users' requests. We compare our transcoding scheduling scheme with an FIFO-based scheme, where the transcoding tasks are performed according to request arrivals in an FIFO manner. For a fair comparison, we assume that users have already been redirected to regions according to our redirection strategy for both schemes. By varying the average computation resource in the regions, we evaluate the fitness of the transcoded segments. In Fig. 14, each sample represents the average bitrate difference between the bitrates of the received version and the requested version at all users versus the average computation resource of the region, calculated as the average number of segments that can be generated by the region. Note that the real computation resource may be different across the regions as it takes different amount of computation resource to transcode different versions. A larger difference indicates a larger streaming quality degradation, as users have to receive a replacement segment with a much smaller bitrate. We observe that the average bitrate difference is much smaller with our design. In particular, our strategy can reduce the number of users who have to receive a segment of a mismatched version by over 42.2%.

4) Replication Cost: Our design utilizes regional preferences of versions when assigning transcoding tasks. Next, we evaluate the replication cost under different numbers of video versions. In Fig. 15, each curve represents the replication cost versus the number of versions, with a particular number of segments in each video. We observe that a larger number of versions leads to a smaller replication cost. The reason is that when more versions are available, our design can effectively allow regions to transcode heterogeneous versions that best meet their users' demand. We also observe that the number of segments has little impact on the replication cost, implying that we can use a small amount of time for adaptive scheduling without incurring increased replication cost. As more and more versions are used in today's adaptive streaming systems, our design reduces not only the waste of computation resource for transcoding, but also the replication cost of the transcoded segments.

V. Related Works

Many architectures have been proposed to implement large-scale video streaming services including the CDN-based architecture [18]. After HTTP has become the norm for users to access online contents, multimedia applications including video streaming, have been largely deployed over HTTP. CDNs can significantly assist in HTTP-based streaming with

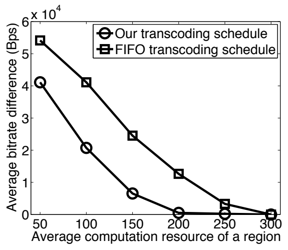


Fig. 14. Comparison of bitrate mismatch under different transcoding schedules.

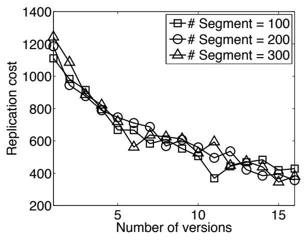


Fig. 15. Average replication cost per time slot versus the number of versions.

servers deployed in multiple geographical locations across multiple ISPs [19]. Users experience higher-quality streaming by receiving streams at more reliable bandwidth from the CDN servers. Recently, Adhikari *et al*. [7] proposed a multi-CDN scheme for real-world video systems to further improve the streaming quality. Traditional studies on video streaming have been focusing on improving the connectivity between streaming servers and users from the network aspect.

Based on the CDN delivery backbone, DASH has recently been proposed to provide adaptive video streaming for heterogeneous networks and devices [1]. Compared with the traditional video streaming paradigm, DASH enables a much larger number of quality versions, requiring a huge amount of computation resource to transcode these versions of videos. Dedicated transcoders are developed to speed up video transcoding [20]. There have also been works on using the computation resource in a cloud cluster for video transcoding. Lao *et al*. [5] designed a MapReduce-based video transcoding scheme for distributing transcoding tasks. Huang *et al*. [13] proposed CloudStream, which schedules the video transcoding tasks inside a cluster according to properties of the videos. Traditional studies on video transcoding have been exploiting dedicated devices or computation resource, leading to the decoupling of segment transcoding and delivery.

Most related works on adaptive streaming have investigated video delivery and video transcoding separately, *i.e.*, videos are pre-transcoded centrally, and then replicated to CDN servers for delivery using a same strategy, *e.g.*, a full replication scheme. In this paper, we explore the design space of joint transcoding and delivery using geo-distributed computation and network resources.

VI. Concluding Remarks

Transcoding and delivery have been separately studied for adaptive video streaming, resulting in a significant waste of computation resource to transcode useless segments, and suboptimal streaming quality due to homogeneous replication of segments of different versions. Motivated by extensive measurement studies, we propose a joint online transcoding and geo-distributed delivery strategy, which allows us to explore a new design space for adaptive video streaming. We connect video transcoding and video delivery based on users' preferences of CDN regions and regional preference of versions to transcode. Aware of users' preferences of CDN regions, our design strategically performs user redirection so that videos can be streamed at large bitrates to the users. Taking into consideration heterogeneous importance of segments and regional preferences of versions to transcode, our design carefully schedules the transcoding tasks so that segments are transcoded to satisfy users' demands in each region, with little need of cross-region replication. Optimization problems are formulated and efficiently solved to derive these strategies. Our trace-driven experiments demonstrate that our design significantly saves computation resource for segment transcoding, improves streaming quality for users, and reduces replication cost for video delivery.

References

[1] I. J. S. W. . (MPEG), "Dynamic adaptive streaming over HTTP," 2010.
[2] A. J. Cahill and C. J. Sreenan, "An Efficient CDN Placement Algorithm for the Delivery of High-Quality TV Content," in *Proc. of ACM Multimedia*, 2004.
[3] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP: Standards and Design Principles," in *Proc. of ACM Conference on Multimedia Systems*, 2011.
[4] Z. Li, Y. Huang, G. Liu, F. Wang, Z. Zhang, and Y. Dai, "Cloud Transcoder: Bridging the Format and Resolution Gap between Internet Videos and Mobile Devices," in *Proc. of ACM NOSSDAV*, 2012.
[5] F. Lao, X. Zhang, and Z. Guo, "Parallelizing Video Transcoding Using Map-Reduce-Based Cloud Computing," in *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2012.
[6] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1357–1370, 2009.
[7] 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 *Proc. of IEEE INFOCOM*, 2012.
[8] Bestv, "http://www.bestv.com.cn/."
[9] Tencent, "http://www.tencent.com/."
[10] J. Cohen, T. Repantis, S. McDermott, S. Smith, and J. Wein, "Keeping Track of 70,000+ Servers: the Akamai Query System," in *Proc. of International Conference on Large Installation System Administration*, 2010.
[11] D. Antoniades, M. Athanatos, A. Papadogiannakis, E. P. Markatos, and C. Dovrolis, "Available Bandwidth Measurement As Simple As Running wget," in *Proc. of Passive and Active Measurement Conference (PAM)*, 2006.
[12] G. P. Zhang, "Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.
[13] Z. Huang, C. Mei, L. Li, and T. Woo, "CloudStream: Delivering High-Quality Streaming Videos Through a Cloud-Based SVC Proxy," in *Proc. of IEEE INFOCOM*, 2011.
[14] Y. Chen, R. H. Katz, and J. D. Kubiatowicz, "Dynamic Replica Placement for Scalable Content Delivery," in *Proc. of International Workshop on Peer-to-Peer Systems*, 2002.
[15] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang, "Challenges, Design and Analysis of a Large-Scale P2P-VoD System," in *Proc. of ACM SIGCOMM*, 2008.
[16] C. Zheng, G. Shen, and S. Li, "Distributed Prefetching Scheme for Random Seek Support in Peer-to-Peer Streaming Applications," in *Proc. of ACM Workshop on Advances in Peer-to-Peer Multimedia Streaming*, 2005.
[17] L. Liang, "The Cloud Video Material Transfer Code System Design in the Global Station Network Environment," in *IEEE International Conference on Image Analysis and Signal Processing (IASP)*, 2012, pp. 1–3.
[18] G. Peng, "CDN: Content Distribution Network," *arXiv preprint cs/0411069*, 2004.
[19] A. Vakali and G. Pallis, "Content Delivery Networks: Status and Trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, 2003.
[20] N. Wu, M. Wen, W. Wu, J. Ren, H. Su, C. Xun, and C. Zhang, "Streaming HD H.264 Encoder on Programmable Processors," in *Proc. of ACM Multimedia*, 2009.