Cloud Transcoder: Bridging the Format and Resolution Gap between Internet Videos and Mobile Devices

Zhenhua Li
EECS, Peking University
Beijing, China
lzh@net.pku.edu.cn

Yan Huang
Tencent Research
Shanghai, China
galehuang@tencent.com

Gang Liu
Tencent Research
Shanghai, China
sinbadliu@tencent.com

Fuchen Wang
Tencent Research
Shanghai, China
futurewang@tencent.com

Zhi-Li Zhang
EECS, University of Minnesota
Minneapolis, MN, US
zhzhang@cs.umn.edu

Yafei Dai
EECS, Peking University
Beijing, China
dyf@pku.edu.cn

ABSTRACT

Despite its increasing popularity, Internet video streaming to mobile devices faces many challenging issues. One such issue is the format and resolution "gap" between Internet videos and mobile devices: many videos available on the Internet are often encoded in formats not supported by users' mobile devices, or in resolutions not best suited for streaming over cellular/WiFi networks. Hence video transcoding to specific devices (and to be streamed over cellular/WiFi networks) is needed. As a computation-intensive task, video transcoding directly on mobile devices is not desirable because of their limited battery capacity. In this paper we propose and implement "Cloud Transcoder" which utilizes an intermediate cloud platform to bridge the format/resolution "gap" by performing video transcoding in the cloud. Specifically, Cloud Transcoder only requires the user to upload a *video request* (i.e., a URL link to the video available on the public Internet as well as the user-specified transcoding parameters) rather than the video content. After getting the video request, Cloud Transcoder downloads the original video from the Internet, transcodes it on the user's demand, and delivers the transcoded video back to the user. Therefore, the mobile device only consumes energy in the last step — but generally with much less energy consumption than downloading the original video from the Internet, due to faster delivery of transcoded video from the "Cloud Transcoder" cloud platform. Running logs of our real-deployed system validate the efficacy of Cloud Transcoder.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Applications

Keywords

Cloud computing, video transcoding, mobile devices

1. INTRODUCTION

Recent years have seen wide adoption of smart mobile devices such as smartphones and tablets. Gartner reports [1] that worldwide sales of mobile devices have far exceeded the PC shipments. Apart from the conventional web surfing, users are increasingly using their mobile devices for Internet video streaming. It is predicted [2] that mobile video traffic will likely dominate the total mobile Internet traffic in the near future. Despite its increasing popularity, Internet video streaming to mobile devices faces many challenging issues, one of which is the format and resolution "gap" between Internet videos and mobile devices.

Due to their relatively small screen sizes but diverse screen resolutions, embedded processors and limited battery capacities, mobile devices usually support a range of video formats and resolutions [3] which are often different from many videos available on the public Internet that are captured and encoded primarily for streaming to desktop and laptop PCs. For example, iPhone4S, one of the most popular and powerful smartphones at present, typically supports MP4 videos up to 640*480 pixels and does not support Adobe Flash videos (FLV). However, today's Internet videos, either uploaded by common users or supplied by large video content providers, are still PC oriented — most videos possess a single format and very limited resolutions. For instance, Youtube usually transcodes its own videos into three resolutions (240p, 360p and 480p) in FLV format in advance to *approximate* its users' devices and bandwidths. As to a mobile device, it often still has to transcode the downloaded video to *match* its specific playback capability, i.e., to *fill the "gap"* between Internet videos and mobile devices.

Because of the aforementioned "gap", there is a growing demand for *video transcoding* so that videos of any format and resolution available on the public Internet can be transcoded *on-demand* to the format and resolution supported by a specific mobile device. One simple solution is to perform video transcoding locally and directly at the user's mobile device (e.g., via a downloadable "video transcoding" app [4, 5] designed for mobile devices). Unfortunately, video transcoding is a highly computation-intensive task — in [6, 7] it is shown that the computation cost of video transcoding is equivalent to that of simultaneously viewing (decoding) multiple videos. Hence locally performing video transcoding on mobile devices will consume a significant amount of

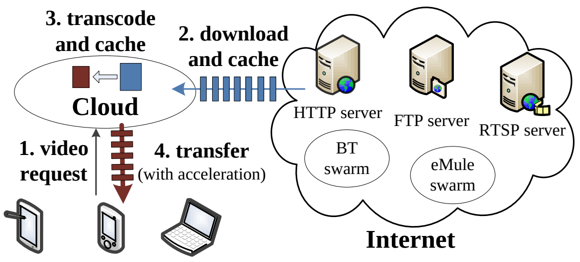


Figure 1: Working principle of Cloud Transcoder.

energy on mobile devices, draining their limited battery capacity. Alternatively, a user may first download the original videos to her PC, utilize specialized video transcoding software (e.g., iTunes or AirVideo [8]) to transcode the videos to the format and resolution supported by her mobile device, and then upload the transcoded videos to her mobile device. Clearly, such an approach is rather inconvenient, and may not always be possible. For instance, when the user is out of office/home, and does not have access to her PC.

The emergence of cloud computing provides a more promising alternative to address this format and resolution "gap" problem: it is natural to imagine using cloud-based video transcoding utilities to perform video transcoding for mobile users on demand. To provide such an *on-demand* video transcoding service, the existing cloud-based transcoding solution (e.g., [9, 10, 11]) typically lets the user upload a video ($\leq$ 100 MB) to the cloud, which subsequently transcodes the original video based on the user's format and resoultion specification, and then delivers the transcoded video back to the user. This solution may work well for transcoding audios and short videos, but is not fit for transcoding *long videos* such as feature-length movies. The main difficulty lies in that it is both time-consuming and energy-consuming for a user to upload a video of long duration (and of higher resolution) to the cloud. This problem is further exacerbated by the asymmetric nature of the Internet access — the uplink bandwidth is generally far lower than the downlink bandwidth, as well as by the fact that the size of the original video is usually larger than that of the transcoded one.

Taking all above into consideration, in this paper we propose and implement "Cloud Transcoder" which utilizes an intermediate cloud platform to bridge the format and resolution gap between Internet videos and mobile devices. As depicted in Figure 1, Cloud Transcoder *only requires* a user to upload a *video (transcoding) request* rather than the video content. The *video request* contains a *video link* and the *user-specified transcoding parameters* including the format, resolution, etc. An example is shown below

$$<\textit{video link}; \textit{format}, \textit{resolution}, \cdots>,$$[1]

where the *video link* can be an HTTP/FTP/RTSP link or a BitTorrent/eMule/Magnet[12] link. After receiving the video request, Cloud Transcoder downloads the original video (says v) using the video link (from the Internet, e.g., an HTTP/FTP/RTSP server or a P2P swarm where the original video is stored) and stores v in the cloud cache, then transcodes v on the user's demand and caches the transcoded video (says v'), and finally transfers v' back to the user *with a high data rate* via the intra-cloud data transfer acceleration. Nowadays, web/P2P download has become the major way in which people obtain video content, so it is reasonable for Cloud Transcoder to require its users to upload their video requests rather than the video content. Therefore, the mobile user only consumes energy in the last step — fast retrieving the transcoded video from the cloud. In a nutshell, Cloud Transcoder provides energy-efficient on-demand video transcoding service to mobile users via its special and practical designs trying to minimize the user-side energy consumption.

Since Cloud Transcoder moves all the video download and transcoding works from its users to the cloud, a critical problem is how to handle the resulting heavy *download bandwidth pressure* and *transcoding computation pressure* on the cloud. To solve this problem, we utilize the *implicit data reuse* among the users and the *explicit transcoding recommendation and prediction techniques*. First of all, for each video v, Cloud Transcoder only downloads it from the Internet when it is requested for the first time, and the subsequent download requests for v are directly satisfied by using its copy in the cloud cache. Such implicit data reuse is completely handled by the cloud and is thus oblivious to users. Meanwhile, the transcoded videos of v (note that v may correspond to multiple transcoded videos in different formats and resolutions which are collectively denoted as $T(v)$) are also stored in the cloud cache to avoid repeated transcoding operations.

Moreover, when a user issues a video (transcoding) request for a video v and the cloud cache has already stored several transcoded versions of v, but the user's transcoding specification does not match any of the existing versions, Cloud Transcoder will *recommend* one of the transcoded videos (with the transcoding parameters closest to what the user has requested) to the user as a possible alternative. If the user accepts the recommended choice, the transcoded video can be delivered to the user immediately. The goal here is to reduce the unnecessary load on the transcoding service; it also speeds up the whole transcoding process for the users. Finally, when the transcoding computation load falls down at night, we also utilize the video popularity based *prediction* to perform video transcoding *in advance*. Namely, we pro-actively transcode the (predicted) most popular videos into a range of formats and resolutions supported by widely held mobile devices, so as to meet the potential user demand and reduce the transcoding load during the peak day time. Based on the real-world performance of Cloud Transcoder, the cache hit rate of the download tasks reaches 87%, while the cache hit rate of the transcoding tasks reaches 66%.

Since May 2011, we have implemented Cloud Transcoder as a novel production system[2] that employs 244 commodity servers deployed across multiple ISPs. It supports popular mobile devices, popular video formats and user-customized video resolutions. Still at its startup stage, Cloud Transcoder receives nearly 8,600 video requests sent from around 4,000 users per day, and 96% of the original videos are long videos ($\geq$ 100 MB). But its system architecture (in particular the cloud cache) is generally designed to serve 100,000 daily requests. Real-system running logs of Cloud Transcoder con-

[1]For a "naive" user who cannot decide what format and resolution he should specify, he can leave the transcoding parameters empty and then Cloud Transcoder will recommend some possible choices to him.

[2]The implementation of Cloud Transcoder is based on a former production system [13] which only downloads videos on behalf of users.

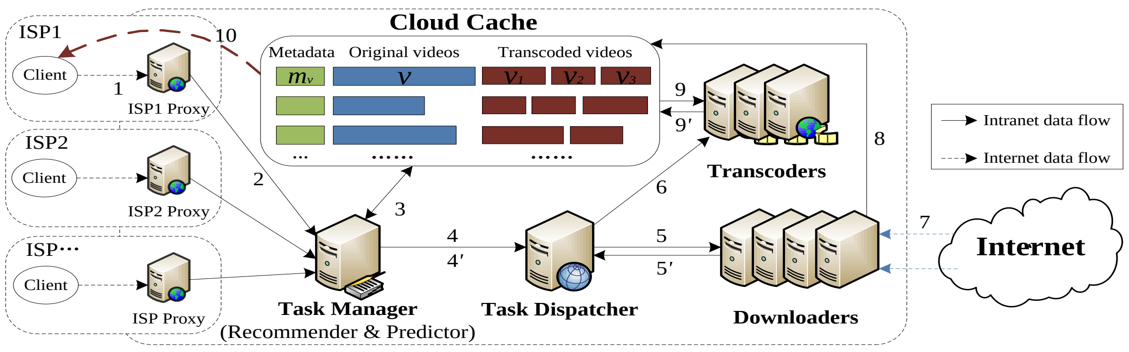


Figure 2: System architecture of Cloud Transcoder.

firm its efficacy. As an average case, a mobile user needs around 33 minutes to retrieve a transcoded video of 466 M-B. The above process typically consumes around 9%/5% of the battery capacity of an iPhone4S/iPad2 ($\approx$ 0.47 WH/1.25 WH, 1 WH = 1 Watt Hour = 3600 J). And the average data transfer rate of transcoded videos reaches 1.9 Mbps, thus enabling the users' view-as-download function. In conclusion, the system architecture, design techniques and real-world performance of Cloud Transcoder provides practical experiences and valuable heuristics to other cloud system designers planning to offer video transcoding service to its users.

The remainder of this paper is organized as follows. Section 2 describes the system design of Cloud Transcoder. Section 3 evaluates the performance of Cloud Transcoder. We discuss about the possible future work in Section 4.

2. SYSTEM DESIGN

2.1 System Overview

The system architecture of Cloud Transcoder is composed of six building blocks: 1) *ISP Proxies*, 2) *Task Manager*, 3) *Task Dispatcher*, 4) *Downloaders*, 5) *Transcoders* and 6) *Cloud Cache*, as plotted in Figure 2. It utilizes 244 commodity servers, including 170 chunk servers making up a 340-TB cloud cache, 20 download servers with 6.5 Gbps of Internet bandwidth, 15 transcoding servers with 60 processing cores @2.4 GHz, 23 upload servers with 6.9 Gbps of Internet bandwidth, etc. Such hardware composition (in particular the cloud cache) is generally designed to serve 100K (K=1000) daily requests, though the current number of daily requests is usually below 10K. Below we describe the organization and working process of Cloud Transcoder by following the message and data flows of a typical video request.

First, a user uploads her video transcoding request to the corresponding ISP Proxy (see Arrow 1 in Figure 2). Each ISP Proxy maintains a *task queue* to control the number of tasks (video requests) sent to the Task Manager (see Arrow 2), so that the Task Manager is resilient to task upsurge in any ISP. Presently, Cloud Transcoder maintains ten ISP Proxies in the ten biggest ISPs in China: Telecom [14], Unicom [15], Mobile [16] and so on. If a user (occasionally) does not locate at any of the ten ISPs, her video request is sent to a random ISP Proxy. On receiving a video request, the Task Manager checks whether the requested video has a copy in the Cloud Cache in two steps (see Arrow 3):

Step 1. Checking the video link. Inside the Cloud Cache, each original video v has a unique hash code and a series of video links pointing to v in its corresponding metadata m_v. If the video link contained in the video request is a P2P link, the Task Manager examines whether the Cloud Cache contains a video that has the same hash code with that contained in the P2P link[3]. Otherwise, the Task Manager directly examines whether the video link is repeated in the Cloud Cache. If the video link is not found, the Task Manager initiates a *video download task* and sends it to the Task Dispatcher (see Arrow 4).

Step 2. Checking the transcoded video. If the video link (pointing to the original video v) is found in the Cloud Cache, the Task Manager further checks whether $T(v)$[4] contains an existing transcoded video that matches the user-customized transcoding parameters. If the requested video actually has a copy, the user can directly and instantly retrieve the video from the Cloud Cache (see Arrow 10). Otherwise, the Task Manager recommends $T(v)$ to the user for a possible choice so as to reduce the transcoding computation pressure on the cloud; thereby, the Task Manager also acts as the "Task Recommender". If the user does not accept any recommendation but insists on her customized transcoding parameters, the Task Manager initiates a *video transcoding task* and sends it to the Task Dispatcher (see Arrow 4').

On receiving a video download task from the Task Manager, the Task Dispatcher assigns the download task to one server (called a "downloader") in the Downloaders (see Arrow 5). For example, if the video link contained in the download task is a P2P link, the assigned downloader will act as a common peer to join the corresponding peer swarm. On receiving a video transcoding task from the Task Manager, the Task Dispatcher assigns the transcoding task to one server (called a "transcoder") in the Transcoders for video transcoding (see Arrow 6).

The Task Dispatcher is mainly responsible for balancing the download bandwidth pressure of the 20 downloaders and the transcoding computation pressure of the 15 transcoders. Each downloader executes multiple download tasks in parallel (see Arrow 7), and the Task Dispatcher always assigns a newly incoming video download task to the downloader with the lightest download bandwidth pressure. Likewise, a newly incoming video transcoding task is always assigned to the transcoder with the lightest computation pressure.

[3]A P2P (BitTorrent/eMule/Magnet) link contains the hash code of its affiliated file in itself, while an HTTP/FTP/RTSP link does not.

[4]$T(v)$ denotes all the existing transcoded videos of v in different formats and resolutions stored in the Cloud Cache.

As long as the downloader accomplishes a download task, it computes the hash code of the downloaded video and attempts to store the video into the Cloud Cache (see Arrow 8). The downloader first checks whether the downloaded video has a copy in the Cloud Cache (using the hash code)[5]. If the video is repeated, the downloader simply discards it. Otherwise, the downloader checks whether the Cloud Cache has enough spare space to store the new video. If the Cloud Cache does not have enough spare space, it deletes some cached videos to get enough spare space to store the new video. The concrete cache capacity planning and cache replacement strategy will be investigated in Section 2.3. When the abovementioned video store process is finished, the downloader notifies the Task Dispatcher (see Arrow 5') for further processing.

When a transcoder receives a video transcoding task, it first reads the corresponding original video from the Cloud Cache (see Arrow 9) and then transcodes it according to the transcoding parameters contained in the video transcoding task. Specifically, each transcoder employs the classic open-source FFmpeg codec software [17] for video transcoding and it supports most popular video formats like MP4, AVI, FLV, WMV and RMVB, as well as user-customized video resolutions. After finishing the transcoding task, the transcoder also checks whether the Cloud Cache has enough spare space to store the new transcoded video (see Arrow 9').

Finally, the requested video is available in the Cloud Cache and the user can retrieve it as she likes (see Arrow 10). Since the user's ISP information can be acquired from her video retrieve message, the Cloud Cache takes advantage of the intra-cloud ISP-aware data upload technique (elaborated in Section 2.4) to accelerate the data transfer process.

2.2 Transcoding Prediction

When the *computation pressure* of the transcoders stays below a certain threshold during a certain period (currently the threshold is empirically set as 50% and the period is set as one hour), the Task Manager starts to predict which videos are likely to be requested for transcoding into which formats and resolutions, based on the video popularity information. The *transcoding computation pressure* is indicated by the *average CPU utilization of the transcoders*. Such prediction often happens at night, when the Task Manager (now behaving as the "Task Predictor") first updates the video popularity information using the user requests received in the latest 24 hours. The video popularity information is two-fold: 1) the number of request times of each video and 2) the most popular transcoding parameters. Currently, the Task Manager picks the top-1000 popular videos and top-3 popular transcoding parameters to initiate transcoding tasks. If a certain popular video has been transcoded into a certain popular format and resolution in the past, the corresponding transcoding task should not be repeated.

Each predicted (non-repeated) transcoding task is sent to the Task Dispatcher to satisfy future potential requests of users, so that part of the transcoding computation pressure in "hot" time has been moved to "cold" time for load balancing (see Figure 3). Besides, we discover the average CPU utilization of the 15 transcoders in the whole day (with prediction) is 34.13%, indicating that 15 transcoders can support at most $\frac{8600}{34.13\%} \approx 25K$ daily requests (8600 is the total number of video requests received in the whole day). Consequently, in order to support 100K daily video requests, we still need to add at least 45 more transcoders in the future.

[5]It is possible that multiple downloaders are downloading the same video content with different video links

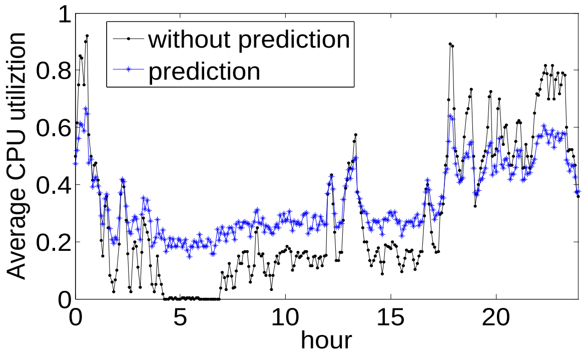


Figure 3: Average CPU utilization of the transcoders in one day (with prediction) and the other day (without prediction), respectively.

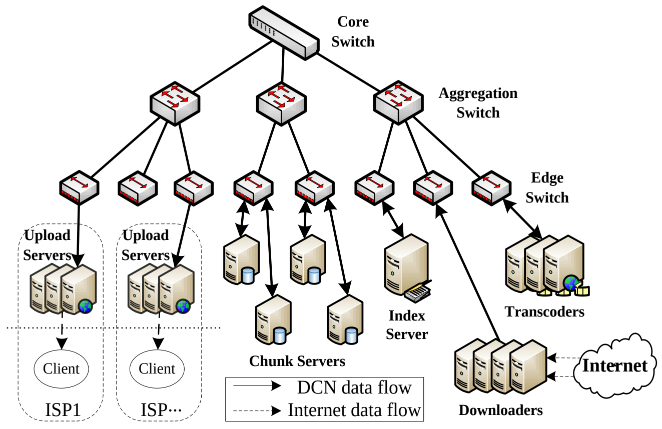
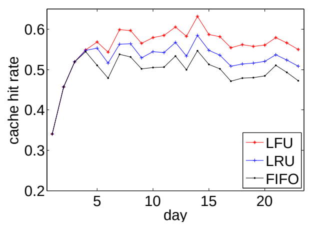


Figure 4: Hardware organization of Cloud Cache.

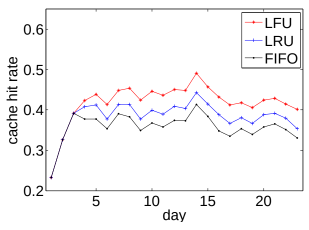
2.3 Cloud Cache Organization

Cloud Cache plays a kernel role in the system architecture of Cloud Transcoder by storing (caching) all the videos and their metadata and meanwhile transferring the transcoded videos back to users. As depicted in Figure 4, the Cloud Cache consists of 170 chunk servers, 23 upload servers and 3 index servers, connected by a DCN (data center network). The DCN adopts the traditional 3-tier tree structure to organize the switches, comprising a core switch in the root of the tree, an aggregation tier in the middle and an edge tier at the leaves of the tree. All the chunk servers, upload servers, index servers, downloaders and transcoders are connected to edge switches. A specific number of *upload servers* are placed in each ISP, approximately proportional to the data traffic volume in each ISP.

Every video (whether original or transcoded) is segmented into chunks of equal size to be stored in the chunk servers, and every chunk has a duplicate for redundancy, so the 170 chunk servers can accommodate a total of $C = \frac{170 \times 4 \text{ TB}}{2} =$ 340 TB unique data. In order to achieve load balance and exploit the chunk-correlation in the same file, all the chunks of a video are stored together into the chunk server with the biggest available storage capacity. The duplicate chunks of



(a) For original videos.



(b) For transcoded videos.

Figure 5: Cache hit rates in simulations.

a video must be stored in another chunk server. There is an index server (as well as two backup index servers) which maintains the metadata of videos.

The 340-TB cloud cache accommodates both original and transcoded videos. Below we first present the cache capacity planning and then discuss the cache replacement strategies. Cloud Transcoder is designed to handle up to 100K daily video requests. Since the average size of original videos is 827 MB (see Figure 7) and every video is stored in the cloud cache for at most 12 days (refer to the user service policy [18]), the total storage capacity of the original videos should be: 827 MB $\times$ 100K $\times$ 12 $=$ 969 TB to well handle 100K daily requests *when there is no data reuse among the users*. According to the running logs of Cloud Transcoder, the current data reuse rate of original videos is about 87% and thus the storage capacity of the original videos is planned as: C_1 $=$ 827 MB $\times$ 100K $\times$ 12 $\times$ $(1-87\%)$ $=$ 126 TB. On the other hand, an original video is associated with three transcoded videos in average and the average size of transcoded videos is 466 MB (see Figure 8), so the storage capacity of the transcoded videos is planned as: $C_2 = 3 \times 466$ MB $\times 100K \times 12 \times (1-87\%) = 213$ TB. As a result, the total cache capacity should be $C = C_1 + C_2 \approx 340$ TB.

Although the current number of daily video requests is much smaller than 100K, it is possible that this number will exceed 100K some day. If the huge upsurge in request number really happens, some data must be replaced to make efficient utilization of the limited cloud cache capacity, where the cache replacement strategy plays a critical role. Here we investigate the performance of the most commonly used cache replacement strategies for both original and transcoded videos via real-trace driven simulations, i.e., FIFO (first in first out), LRU (least recently used) and LFU (least frequently used). The trace is a 23-day system log (refer to Section 3 for detailed information) of all the video requests and the simulated cloud cache storage is 30 TB ($\approx \frac{8600}{100K} \cdot 340$ TB, where 8600 is the current average number of daily requests). From Figure 5 we discover that for both original and transcoded videos, FIFO performs the worst and LFU performs the best to achieve the highest cache hit rate.

2.4 Accelerating the Data Transfer of Transcoded Videos

A critical problem of Cloud Transcoder is how to accelerate the transfer process of the transcoded videos from the cloud to the user in order to save the users' energy consumption in retrieving their requested videos. Considering that the cross-ISP data transfer performance degrades seriously and the inter-ISP traffic cost is often expensive, we solve this problem via the *intra-cloud ISP-aware data upload technique*. Since the user's real-time ISP information can be acquired from her video retrieve message, the Cloud Cache takes advantage of its ISP-aware upload servers to restrict the data transfer path within the same ISP as the user's, so as to enhance the data transfer rate and avoid inter-ISP traffic cost. Specifically, suppose a user A locating at ISP1 wants to retrieve a video v' stored in a chunk server S, and the Cloud Cache has placed 3 upload servers (U_1, U_2 and U_3) in ISP1 (see Figure 4). The chunks of v' are first transferred from S to a random upload server (says U_3) in ISP1, and then transferred from U_3 to the user A. The transfer process is not store-and-forward but pass-through: as soon as U_3 gets a complete chunk of v', U_3 transfers the chunk to A. v' would not be cached in U_3 because the intra-cloud end-to-end bandwidth is quite high (1 Gbps) and we do not want to make things unnecessarily complicated.

3. PERFORMANCE EVALUATION

We use the complete running log of Cloud Transcoder in 23 days (Oct. 1–23, 2011) to evaluate its performance. The log includes the performance information of 197,400 video transcoding tasks involving 76,293 unique videos. The daily statistics are plotted in Figure 6. For each task, we record its *user device type*, *video link*, *transcoding parameters*, *original size*, *transcoded size*, *download duration time* (of the cloud downloader), *transcoding time*, *retrieve duration time* (of the user) and so on. 85% of the video links sent from users are P2P links. The most popular transcoding parameters include MP4-1024*768 (10%, mostly coming from iPad users), MP4-640*480 (38%, mostly coming from iPhone and Android smartphone users) and 3GP-352*288 (27%, mostly coming from Android smartphone users). As shown in Figure 7 and Figure 8, the average file size of the original videos is 827 MB, as 1.77 times as that of the transcoded videos (466 MB). 96% of the original videos are long videos ($\geq$ 100 MB).

As an average case, a mobile user needs around 33 minutes to retrieve a transcoded video (see Figure 9) with the help of the intra-cloud data transfer acceleration. The above process may consume 6.1%/5.5% of the battery capacity of an iPhone4S/iPad2 *in theory*, given that the battery of iPhone4S/iPad2 is claimed to support about 9/10 hours' WiFi data transfer. To check the practical energy consumption, we use our own iPhones/iPads to retrieve a long enough transcoded video from Cloud Transcoder (via WiFi) for 33 minutes and then record their battery consumptions. All the other user applications are closed, and the screen brightness is set to 25% with "auto-adjustment" disabled. The results are listed in the following table, indicating that the iPhone battery consumption is typically around 9% ($\approx$ 0.47 WH) while the iPad battery consumption is typically around 5% ($\approx$ 1.25 WH).

Data transfer rate (≈KBps)	50	100	200	300
iPhone battery consumption (%)	8.7	8.9	9.0	9.2
iPad2 battery consumption (%)	4.5	4.8	5.0	5.1

As a contrast, Figure 10 illustrates the average download duration time for a downloader (in Cloud Transcoder) to get an original video is 189 minutes (as 5.72 times as the average retrieve duration time), because each downloader directly gets data from the Internet in the common way (without designated acceleration). Finally, Figure 11 indicates that

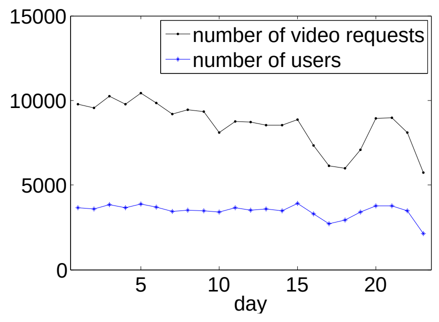


Figure 6: Daily statistics.

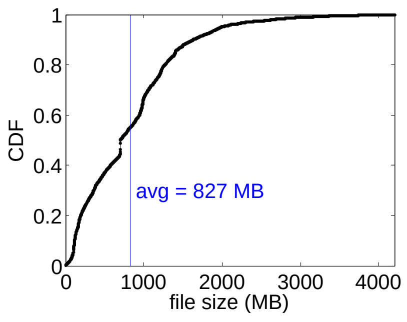


Figure 7: Original file size.

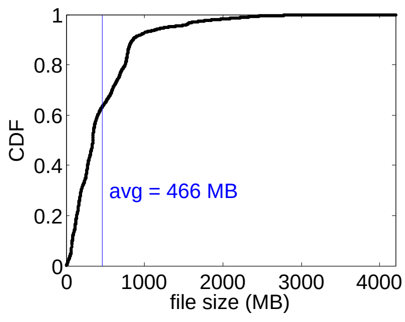


Figure 8: Transcoded file size.

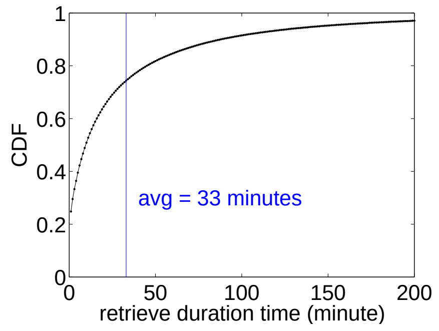


Figure 9: Retrieve duration.

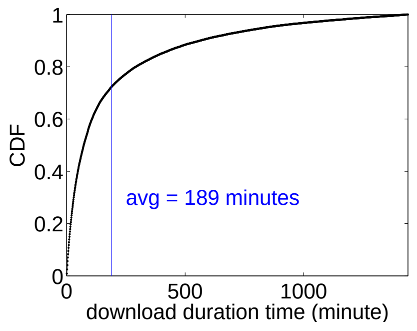


Figure 10: Download duration.

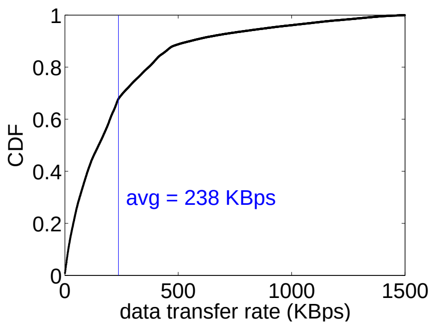


Figure 11: Data transfer rate.

the average data transfer rate of transcoded videos reaches 238 KBps (= 1.9 Mbps), thus enabling the users' view-as-download function.

4. FUTURE WORK

Still some future work remains. First, as a novel production system still at its startup stage, Cloud Transcoder tends to adopt "straightforward and solid" designs in constructing each component so that the deployment and debugging works are easy to handle. We realize there is still considerable optimization space for better design to take effect and this paper is the first step of our efforts.

Second, some web browsers also start to provide video transcoding service. For example, UCWeb [19], the most popular mobile web browser in China, has employed cloud utilities to transcode web flash videos into three resolutions: 144*176, 176*208 and 240*320, in order to facilitate its mobile users. Amazon has recently claimed that its novel Silk web browser [20] will transcode Internet videos to certain formats and resolutions (especially fit for its 7-inch Kindle Fire Tablet) by using its EC2 cloud platform. We have begun to explore integrating the service of Cloud Transcoder to the QQ web browser [21].

5. ACKNOWLEDGEMENTS

This work is supported in part by the China "973" Grant. 2011CB302305, China NSF Grant. 61073015, and US NSF Grants CNS-0905037, CNS-1017647 and CNS-1117536.

6. REFERENCES

[1] http://www.gartner.com/it/page.jsp?id=1848514.
[2] Cisco traffic report. http://www.cisco.com/en/US /solutions/collateral/ns341/ns525/ns537/ns705/ns827 /white_paper_c11-520862.pdf.
[3] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen. "A Server's Perspective of Internet Streaming Delivery to Mobile Devices," IEEE INFOCOM, 2012.
[4] Android Media Converter. https://market.android.com/details?id=com.ghostmobile.mediaconverter.
[5] Android VLC Pro. https://market.android.com/ details?id=com.gmail.traveldevel.android.vlc.license.
[6] J. Ostermann, et al. "Video coding with H.264 /AVC: tools, performance and complexity," IEEE Circuits and Systems magazine, vol. 4, no. 1, 2004.
[7] Z. Huang, C. Mei, L. Li, and T. Woo. "CloudStream: Delivering high-quality streaming videos through a cloud-based SVC proxy," IEEE INFOCOM, 2011.
[8] http://itunes.apple.com/app/id306550020.
[9] YouConvertIt. http://www.youconvertit.com.
[10] Online-convert. http://www.online-convert.com.
[11] Mov-avi. http://online.movavi.com.
[12] http://en.wikipedia.org/wiki/Magnet_URI_scheme.
[13] Y. Huang, Z. Li, G. Liu, and Y. Dai. "Cloud Download: Using Cloud Utilities to Achieve High-quality Content Distribution for Unpopular Videos," ACM Multimedia, 2011.
[14] China Telecom. http://www.chinatelecom.com.cn.
[15] China Unicom. http://www.chinaunicom.com.cn.
[16] China Mobile. http://www.10086.cn.
[17] FFmpeg web site. http://ffmpeg.org.
[18] http://xf.qq.com/help_video.html.
[19] UCWeb browser. http://www.ucweb.com.
[20] Amazon Silk. http://amazonsilk.wordpress.com.
[21] QQ web browser. http://browser.qq.com.