Public and open HEVC encoding service in the cloud

Public and Open HEVC Encoding Service in the Cloud

Aaro Altonen, Marko Viitanen, Joni Räsänen, Alexandre Mercat and Jarno Vanne

Tampere University
Korkeakoulunkatu 10, Tampere, 33720, Finland
{aaro.altonen, marko.viitanen, joni.rasanen, alexandre.mercat, jarno.vanne}@tuni.fi

ABSTRACT

The ability to record vast amounts of video content requires convenient and efficient video coding services with which users can tackle the limited storage and transmission capacities. This paper presents an open-source cloud service for encoding raw video formats and transcoding compressed videos to the latest HEVC/H.265 format. Respective commercial transcoding services are available on the Internet but they are behind a paywall. On the other hand, using command-line interfaces of existing open-source software solutions requires in-depth knowledge of the coding process to attain the best coding gain and speed. The proposed service is available online, it is free to use without any registration, and its easy-to-use web interface makes it feasible for non-technical users. It is built on the FFmpeg multimedia framework whose built-in decoders accept various input video formats that are then compressed to HEVC with a full-fledged Kvazaar open-source encoder.

CCS CONCEPTS

• **Information systems** → World Wide Web • **Networks** → Cloud computing • **Software and its engineering** → Open source model

KEYWORDS

High Efficiency Video Coding (HEVC), Kvazaar HEVC encoder, FFmpeg, Software as a Service (SaaS), Cloud en/transcoding

ACM Reference format:

Aaro Altonen, Marko Viitanen, Joni Räsänen, Alexandre Mercat, and Jarno Vanne. 2019. Public and Open HEVC Encoding Service in the Cloud. In *Proceedings of ACM Multimedia Systems (MMSys'19). ACM, Amherst, MA, USA, 4 pages.* https://doi.org/10.1145/3304109.3323834

1 Introduction

High Efficiency Video Coding (*HEVC/H.265*) [1] is the current state-of-the-art video coding standard. It is targeted to reduce bit rate by 50% over the preceding MPEG AVC/H.264 standard [2] for the same subjective visual quality [3]. Decreasing file size by half is enticing to anyone recording, processing, or streaming video. However, manually converting video files to HEVC is cumbersome and tends to require profound understanding of video codecs, making HEVC coding unfeasible for a non-technical user.

The latest cloud transcoding services bring video format conversion to a wider public by hiding most of the technical details from the user. The most advanced services are able to encode raw videos and transcode a wide selection of compressed formats to a desired output format. In general, video transcoding [4] can be implemented by either direct digital-to-digital conversion from one format to another to attain the fastest speed or it can be composed of separate decoding and encoding stages. The latter solution has gained more popularity since it is more modular and tolerant to different input and output video formats.

Table 1 lists existing transcoder solutions classified according to their features. The systems are compared in terms of pricing models, availability of the source code, and encoding capabilities. Some of the open-source encoders are completely unavailable and so is their pricing. The most well-known closed solutions are hosted by Amazon [5], Coconut [6], Qencode [7], and Zencoder [8]. However, these commercial services are all behind a paywall, typically with a per-minute pricing model. There also exist a few academic approaches. Z. H. Chang et al. [9] implemented a real-time distributed system for streamed video transcoding, M. Chen et al. [10] worked on parallel transcoding, and Y. Dong et al. [11] introduced a containerized cloud transcoding service for easier deployment. These systems are, however, not available for public and their technical details are not reported. Therefore, they are excluded from Table 1.

The prior-art open-source solutions include Cloud Transcode [12], Morph [13], and Snickers [14], of which the latter two are not maintained actively anymore. Cloud Transcode and Snickers support only direct uploads via HTTP-link or S3. Both of them are also tied to Amazon S3 and these services are only accessible through an *application programming interface* (*API*). Morph is limited to MP4 input and AVC output formats. Furthermore, they are all lacking an easy-to-use *user interface* (*UI*).

This paper describes our Kvazaar cloud encoding service [15] for encoding raw input videos and transcoding all popular video formats to HEVC. Unlike all other listed solutions, our service is

Table 1: Comparison of existing cloud encoding services

	Free	*Open*	*Raw Input*	*HEVC Output*
Amazon [5]	no	no	yes	yes
Coconut [6]	no	no	no	yes
Qencode [7]	no	no	no	yes
Zencoder [8]	no	no	yes	yes
Cloud Transcode [12]	n/a	yes	no	yes
Morph [13]	n/a	yes	no	no
Snickers [14]	n/a	yes	no	yes
***Kvazaar Cloud** [15]*	yes	yes	yes	yes

free to use, open source, and easily deployable. It offers a user-friendly UI for uploading input and downloading output files.

This paper is organized as follows. Section 2 gives an overview of the applied encoder software. Section 3 takes an in-depth look at the proposed cloud encoding architecture. Section 4 describes the entire processing flow of Kvazaar Cloud Encoder. Section 5 describes the demo setup and Section 6 concludes the paper.

2 Kvazaar HEVC Encoder

The proposed cloud encoding service is powered by the award-winning Kvazaar open-source HEVC encoder [16], [17]. It is being developed by Ultra Video Group at Tampere University and it is available with the LGPL 2.1 license.

Kvazaar makes real-time 4K encoding possible with the ultrafast setting whereas it reaches coding efficiency close to that of HEVC reference encoder [18] with the veryslow preset. Kvazaar supports almost all HEVC coding tools including Wavefront Parallel Processing, Overlapped WaveFront, multithreading, deblocking filter, Sample Adaptive Offset, sub-pixel motion estimation, prediction unit depth limitation, bi-prediction, and rate control. The most demanding functions are optimized using SSE4.1 and AVX2 instruction sets for x86/x64. Most of the coding tools are controlled via the *command-line interface* (*CLI*) or with the API, if Kvazaar library is used. Supported input format is YUV 4:2:0 and output conforms to Main or Main 10 HEVC standard profiles.

3 Proposed Kvazaar Cloud Encoding System

The proposed system is implemented in Node.js/Javascript. Besides Kvazaar, it utilizes many open-source software components such as Docker [19], FFmpeg [20], PostgreSQL [21], and Redis [22].

Figure 1 presents the system components with their main relations and interactions. A communication between a Browser Client and a Node.js Server is implemented with two public interfaces: WebSocket and regular HTTP. The WebSocket is responsible for relaying control messages from the server to the client. The media file upload and the web UI are transferred through the HTTP. Public API access is not provided.

The Node.js Server is composed of the five parts: 1) Controller, 2) Parameter Parser, 3) Queue Manager, 4) Storage, and 5) Worker.

Controller interfaces the WebSocket requests and the rest of the system. It validates the requests using the Parameter Parser and signals the progress to the client. Controller is also responsible for access control to the storage.

Parameter Parser validates the selected processing parameters. Each parameter has a list of suitable values which are used in validation. Different validation types are used, e.g., pixel formats are validated using a string look-up table, whereas the encoding parameter values are strings, boolean, or regular expression patterns.

Queue Manager communicates with the Worker threads and takes care of the task allocation in the system. The system uses Redis in-memory key-value database to handle the message/task queue data exchange.

Storage is the virtual component mapped to the physical hardware. It is used to store all media and intermediate files under processing. It also contains the PostgreSQL and Redis databases, which are used by Controller, Worker, and Queue Manager.

Worker is the major actor in the system. It processes the video en/transcoding tasks according to received parameters. It includes FFmpeg and Kvazaar for media processing and PostgreSQL for metadata storage. FFmpeg performs the initial file parsing, audio extraction, video decoding, and muxing of the final container. Kvazaar is used for HEVC encoding according to the given encoding parameters. Worker keeps track of the progress of the task, allowing cancellation of an ongoing task upon request. A predefined number of Worker processes are spawned when starting the server.

The whole system is wrapped inside a Docker container, which allows dynamic deployment of the system for private use and creates an additional layer of security against malicious actors.

The system can be used in two ways: either by using the version hosted by the Ultra Video Group at **http://ultravideo.cs.tut.fi/cloud** or downloading the provided sources with Dockerfile and hosting it locally.

4 Processing Flow of Kvazaar Cloud Encoder

Kvazaar Cloud Encoder is designed to serve both regular and advanced users. It provides an easy-to-use UI with default settings but it also allows for fine-grained control over the coding process.

4.1 Uploading and Validation

Figure 2 and 3 show the main view of the UI and the advanced settings menu of the Kvazaar Cloud Encoder, respectively. A video file is first selected for processing. The system accepts the main input formats and codecs supported by FFmpeg. The format can be raw or containerized video. The encoder options can be automatically set using encoding levels (Figure 2), which offer a

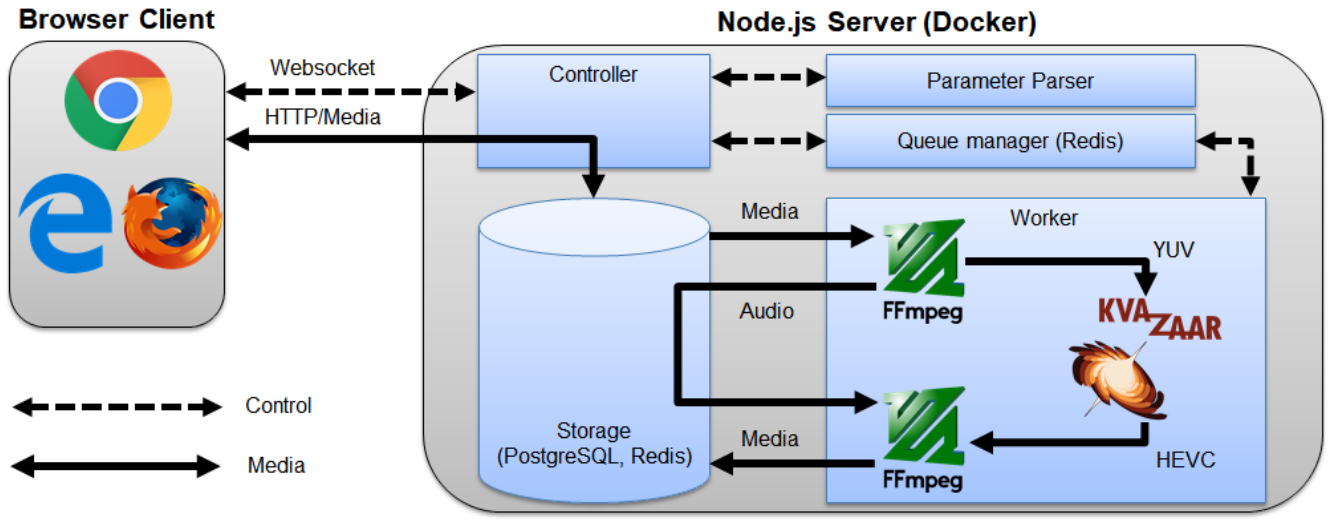

Figure 1: Overview of Kvazaar Cloud Encoder process

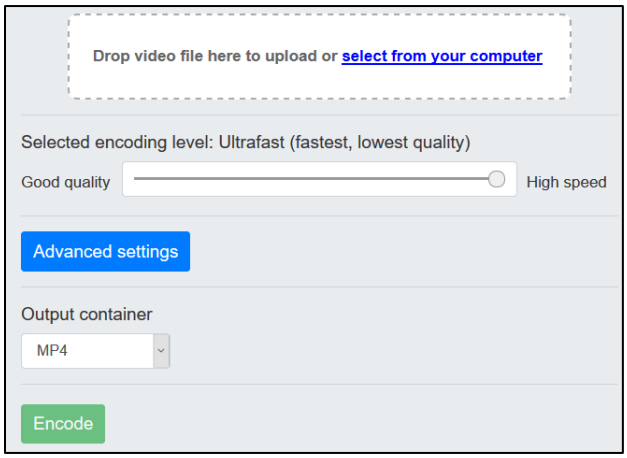

Figure 2: Main view of the UI

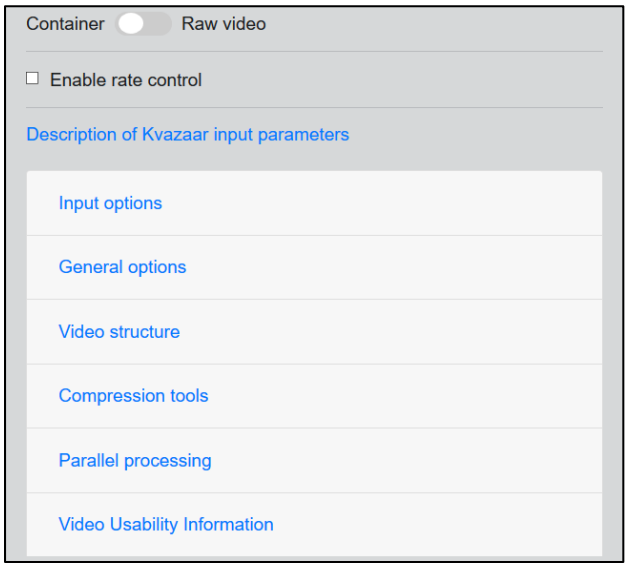

Figure 3: Advanced Kvazaar settings menu

trade-off between encoding speed and quality. In addition, the parameters can be adjusted manually using the advanced settings menu (Figure 3). For a raw video, the input format, video dimensions, and frame rate have to be specified.

The request to upload a video is validated by the server using an *uploadRequest* via WebSocket to the Controller. The server then parses the request, validates all sent options, and either approves or rejects the file upload in *uploadResponse* message.

The upload starts when the request is accepted. A file is uploaded in 5 MB chunks. If the total amount of uploaded bytes exceeds the predefined limit, the server stops the upload and informs a client that the file size has been exceeded. The server also extracts the duration of the video either from the header data, or from the file size in the case of a raw format. For the processing to continue, the duration must be extracted successfully and the video length must be under a predefined limit.

A client is notified about the upload completion and the request is dispatched to a worker thread which starts processing the video. If all worker threads are busy, the task is put into the queue and the client is notified.

4.2 Encoding/Transcoding

In the case of a raw input, the running workers first convert the video into YUV 4:2:0 format (if not already) and start the encoding process immediately. Otherwise, the workers first decode the input video file using FFmpeg after which the file is encoded using an instance of Kvazaar. Status updates of the progress are transmitted to the client via the WebSocket.

Cancelling an ongoing task is possible in the UI, which sends *cancelRequest* via WebSocket to the Controller. The server first validates and then cancels the request by killing the associated process or by removing the task from the queue. It sends a *cancelResponse* back to the client indicating that the response was received and processed.

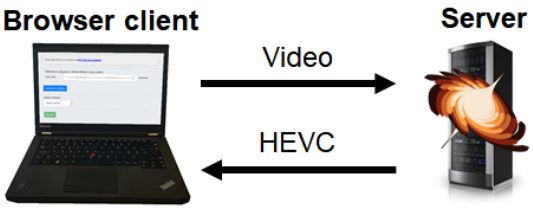

Figure 4: High-level system setup of the demonstration

4.3 Muxing and Downloading

After encoding, muxing is done if an output file container is requested. The worker muxes the video using FFmpeg and includes the audio track of the original file in the output container.

Finally, the client is notified via the active WebSocket connection that the video is ready for download. The service is intended for video encoding only, so a limit of two downloads is set to prevent intentional file sharing. The file is removed from the server after two downloads. An interrupted download is also counted in.

5 Demonstration Setup

The demonstrator at the conference site includes a laptop that is connected to the Internet. The proposed service is used through a public web interface but the server is also running in the laptop for local demonstrations. A high-level depiction of the setup is shown in Figure 4.

A user can select an input video for the service from the laptop by either dragging it to the upload form or selecting it manually using the open file dialog. Using the encoding level option (Figure 2), the user can select a trade-off between coding speed and quality. The encoding process can also be customized by selecting parameters manually in the advanced settings menu (Figure 3).

Pressing the *Encode* button triggers the client to send a validation request to the server. If the input file is valid, the file upload is started. A progress bar shows the upload status. When ready, the user can change to *My videos* view and follow the encoding process. User can interrupt the request anytime by clicking the *Cancel* button. After the coding task is completed, the coding state is changed to *Ready*. The user can then download the HEVC file from the server by clicking the *Download* button and play the video in the media player. Alternatively, the file can be deleted with the *Delete* button.

During the demonstration, up to five encoding processes can be run on the server simultaneously. Each encoding process can be visualized with an associated flow diagram, which gives an in-depth view of the processing steps on the server. Each flow diagram is updated dynamically by highlighting the executed step and annotating it with a detailed description. The CPU load and the memory usage can also be monitored in real time and the most demanding tasks are identified in the flow diagram(s). Furthermore, the console prints of the backend show the info extracted from each video as well as the input signals given by the user via the UI.

6 Conclusions

The explosive growth of video sparks a need for efficient HEVC compression but there are no easy-to-use and free HEVC coding tools. This paper introduced a user-friendly Kvazaar cloud encoding service for converting raw or obsolete video formats to HEVC. Our proposal is built on FFmpeg and award-winning Kvazaar encoder is utilized to take care of HEVC encoding inside the service. The proposed service is free, easy to use, and available as open-source for anyone to set up. It also provides in-depth customization of the encoding process by allowing the user to select the main encoder parameters of Kvazaar. Future objective is to assign predefined coding parameters for specific tasks, such as 360-degree video coding. The possibility of live streaming will also be studied.

ACKNOWLEDGMENTS

This work was supported in part by the European Celtic-Plus Project VIRTUOSE and the Academy of Finland (decision no. 301820). The authors would also like to thank all contributors of the Kvazaar open-source project [16].

REFERENCES

[1] *High Efficiency Video Coding*, document ITU-T Rec. H.265 and ISO/IEC 23008-2 (HEVC), ITU-T and ISO/IEC, Apr. 2013.
[2] *Advanced Video Coding for Generic Audiovisual Services*, document ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC), ITU-T and ISO/IEC, Mar. 2009.
[3] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards—Including high efficiency video coding (HEVC)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1669-1684, Dec. 2012.
[4] I. Ahmad, X. Wei, Y. Sun, and Y. Q. Zhang, "Video transcoding: an overview of various techniques and research issues," *IEEE Trans. Multimedia*, vol. 7, no. 5, pp. 793-804, Oct. 2005.
[5] *Amazon Elastic Transcoder* [Online]. Available: https://aws.amazon.com/elastictranscoder/
[6] *Coconut* [Online]. Available: https://coconut.co/
[7] *Qencode* [Online]. Available: https://cloud.qencode.com/
[8] *Brightcove Zencoder* [Online] Available: https://zencoder.com/en/
[9] Z. H. Chang, B. F. Jong, W. J. Wong, and M. D. Wong, "Distributed video transcoding on a heterogeneous computing platform," *in Proc. IEEE Asia Pacific Conf. Circuits Syst.*, Jeju, South Korea, Oct. 2016.
[10] M. Chen, W. Chen, Z. Liu, and L. Cai, "Parallel video transcoding using Hadoop MapReduce," *Journal of Network Computing and Applications*, vol. 1, pp. 7-11, 2016.
[11] Y. Dong, X. Zhang, Y. Zhao, and L. Song, "A containerized media cloud for video transcoding service," *in Proc. IEEE Int. Conf. Consumer Electron*., Las Vegas, NV, USA, Jan. 2018.
[12] *CloudTranscode* [Online]. Available: https://github.com/bfansports/CloudTranscode
[13] G. Gao and Y. Wen. "Morph: a fast and scalable cloud transcoding system," in *Proc. ACM Int. Conf. Multimedia,* Amsterdam, The Netherlands, Oct. 2016. DOI: https://doi.org/10.1145/2964284.2973792
[14] *Snickers Video Encoder* [Online]. Available: https://github.com/snickers/snickers
[15] *Kvazaar Cloud Encoder* [Online]. Available: https://github.com/ultravideo/cloud-encoder
[16] *Kvazaar HEVC Encoder* [Online] Available: https://github.com/ultravideo/kvazaar
[17] M. Viitanen, A. Koivula, A. Lemmetti, A. Ylä-Outinen, J. Vanne, and T. D. Hämäläinen, "Kvazaar: open-source HEVC/H.265 encoder," *in Proc. ACM Int. Conf. Multimedia,* Amsterdam, The Netherlands, Oct. 2016. DOI: https://doi.org/10.1145/2964284.2973796
[18] *Joint Collaborative Team on Video Coding Reference Software*, HM [Online]. Available: http://hevc.hhi.fraunhofer.de/
[19] *Docker* [Online]. Available: https://www.docker.com/
[20] *FFmpeg* [Online]. Available: https://www.ffmpeg.org/
[21] *PostgreSQL* [Online]. Available: https://www.postgresql.org/
[22] *Redis* [Online]. Available: https://github.com/antirez/redis