Volume XXV 2022
ISSUE no.2
MBNA Publishing House Constanta 2022

Scientific Bulletin of Naval Academy

SBNA PAPER • OPEN ACCESS

Optimization of video recording pipeline for open-source conferencing software - a case study on BigBlueButton

To cite this article: A. Atodiresei, E. Băutu and A. Băutu, Scientific Bulletin of Naval Academy, Vol. XXV 2022, pg. 169-176.

Submitted: 08.03.2023
Revised: 15.03.2023
Accepted: 20.03.2023

Available online at www.anmb.ro

ISSN: 2392-8956; ISSN-L: 1454-864X

doi: 10.21279/1454-864X-22-I2-016

Optimization of video recording pipeline for open-source conferencing software - a case study on BigBlueButton

Anca Atodiresei[1], Elena Băutu[2], and Andrei Băutu[3]

[1] "Mircea cel Bătrân" Naval Academy, Romania, anca.atodiresei@anmb.ro
[2] "Ovidius" University, Romania, ebautu@univ-ovidius.ro
[3] "Mircea cel Bătrân" Naval Academy, Romania, andrei.bautu@anmb.ro

Abstract. Due to the Covid-19 pandemic, most entities from the industry, finance, social, and education fields had moved their daily office meetings and classes to online video conferencing platforms, to avoid the spread of the virus, but still, they want to maintain the greatest degree of human interaction possible, which raised the demand for paid and free online video conferencing services. Due to the rising demand, the pressure on communication networks and data centers processing video streams, both live and recordings, has increased exponentially, to the point of causing service outages. To prevent such problems, officials made public appeals to companies and end-users to monitor and reduce their video streaming demands, tech companies are investing in extending their data center capabilities, and researchers and software companies are looking for ways to optimize the performance of existing systems concerning video processing and transmission. In this paper, we analyse and optimize the processing pipeline for video recordings of online meetings held via Bigbluebutton, a well-known open-source software for videoconferencing. The main goal of our research is to reduce the computational load of its processing pipeline, while maintaining content quality and, if possible, also reduce bandwidth requirements.

1. Introduction

In 2020, the outbreak of the coronavirus pandemic challenged our traditional ways of living and working, forcing us to reduce direct human interactions to minimum levels. In this context, traditional face-to-face communication methods in businesses, education, public administration, entertainment, etc. were forcibly replaced with online communication.

Many organizations, companies, in particular, were already in the process of shifting their communication towards online and this pandemic only accelerated their efforts to do so. For other, more traditional organizations, like public institutions and universities, face-to-face communication was deeply rooted inside their operational core. In their case, the shift to online communication caused a shockwave within the organization that threatened to bring their operation to a halt.

In the case of educational institutions (but also for other organizations, like theaters, charities, non-profit groups, etc.), big tech companies stepped up to help them by temporarily removing or raising the usage limits for their free services. Microsoft and Google released new versions of their business products (Teams and Classroom) with increased conferencing capacities, Zoom and Cisco removed the time limit for their free conferencing products when used for academic purposes. However, as time passed, the tech companies had to return to their original business model and the educational institutions had to choose between adapting to the new limits, signing up for paid services, or deploy their infrastructure for online conferencing using open-source software.

Luckily, the open-source community offers quite a few of highly mature projects for online conferencing, like BigBlueButton, Jitsy Meet, Jami, eduMeet. Therefore, many companies and universities decided to try to host their own online video conferencing solutions. For them, managing their own video-conferencing platform has some disadvantages: maintenance costs of the IT infrastructure (but in many cases, these are lower than a medium-size premium subscription for the same services), requirements for highly skilled system administrators, the need of constant monitoring, etc. But there are also many advantages: no limitations and no costs with paid subscriptions for conferencing services, data confidentiality, integration flexibility with other systems (like Content Management Systems and Learning Management Systems), many opportunities for customization and branding.

One option that universities, in particular, demand is the ability to record the online conferences for archiving purposes, for legal purposes, or publication within their Learning Management Systems. BigBlueButton is one of the free open-source videoconferencing systems that have such features [1][2][3][4][5][6][7]. It was designed as a tool for online learning and it can be easily integrated with many learning and content management systems (Moodle, Sakai, Ilias, Canvas, Drupal, Workpress, etc). In Romania, BigBlueButton is used for online classes in many universities, like Transylvania University of Brașov, Danubius University, "Mircea cel Batran" Naval Academy, and "Ovidius" University of Constanța.

For performance reasons, while a meeting is on-going, BigBlueButton stores content (video, audio, presentations, etc) used by it in raw format. Once the meeting ends, this content enters a processing pipeline that will eventually transform it into a recording of the conference which can be managed and published online via an REST API.

In this paper we discuss how the post-meeting processing pipeline for video recordings in BigBlueButton works, we identify some performance bottlenecks, present solutions for fixing them, and assess their impact concerning both processing speed, but also output quality.

2. Architecture of BigBlueButton post-meeting processing pipeline

By default, during conference meetings, BigBlueButton records all audio, webcam videos, desktop sharing video, chats, slides, and whiteboard events for later processing. Storing all this information during conferences allows for processing pipelines, after the meeting ends, to access the information gathered and generate meeting recordings in various presentation formats.

After each meeting ends, the raw meeting content enters the processing pipeline described in Figure 1, which contains the following steps: archive, sanity check, process, and publish.

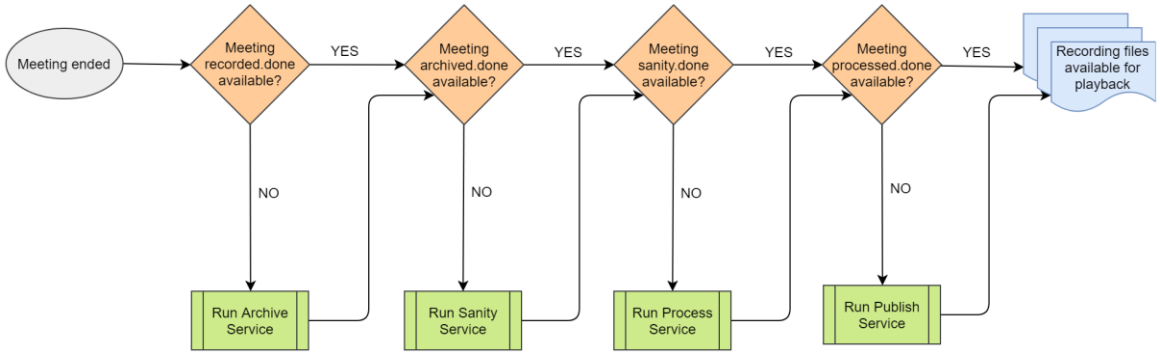

Figure 1. The stages of the processing pipeline for meeting recordings [8]

The Archive stage of the processing pipeline gathers all files related to the meeting, generated by various systems, and copies them to a single directory. This process also checks if the recording should be processed (i.e. the meeting moderator enabled the "Record" feature) or if it should be deleted. The steps performed during the archive stage are described in Figure 2.

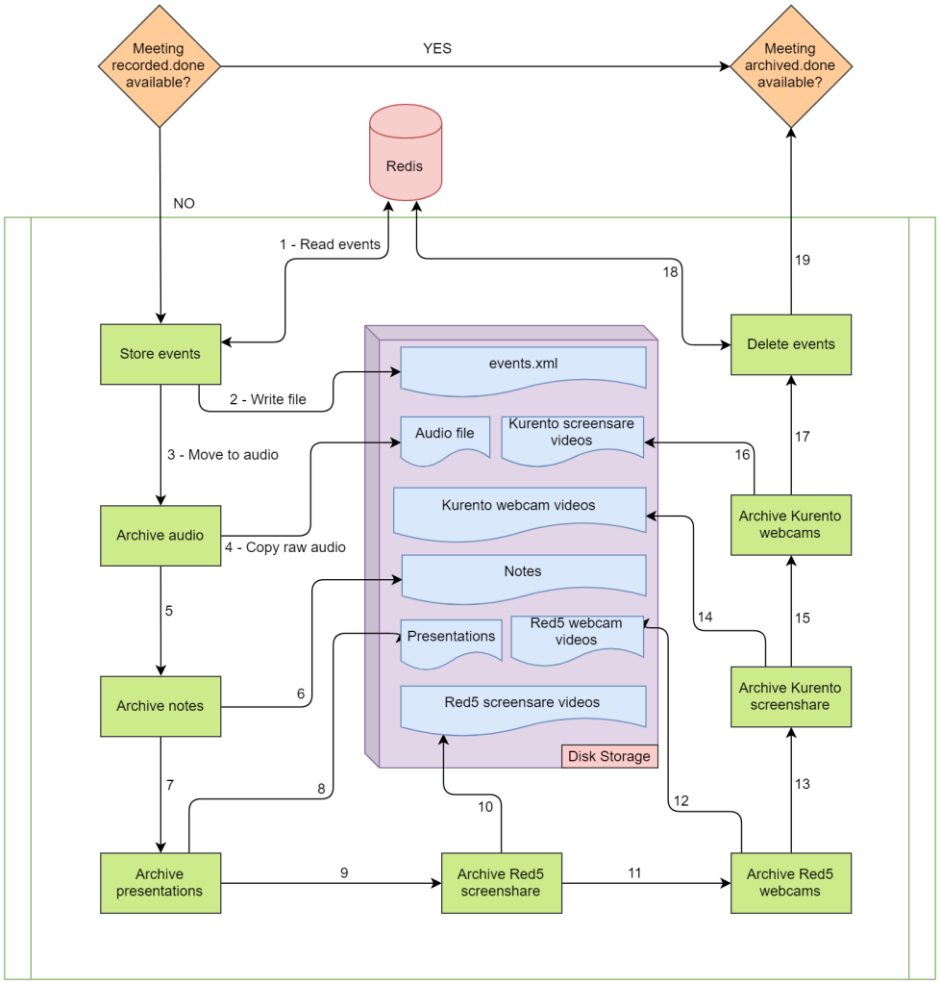

Figure 2. The flowchart of the Archive stage of BigBlueButton [8]

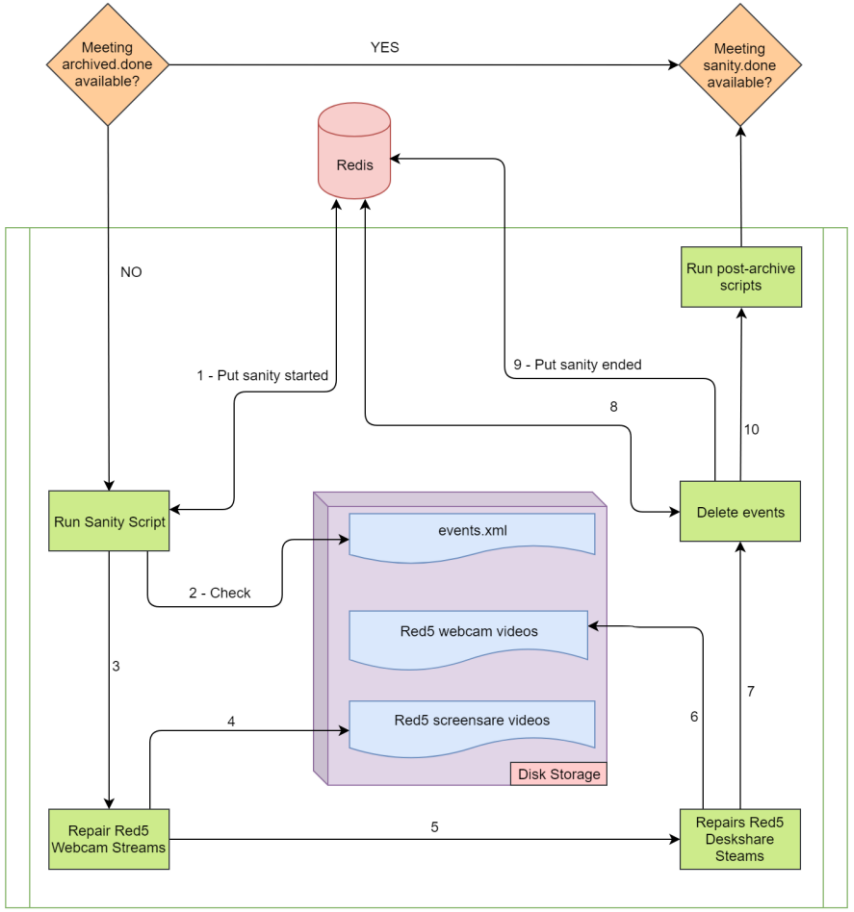

Figure 3. The flowchart of the Sanity check stage of BigBlueButton [8]

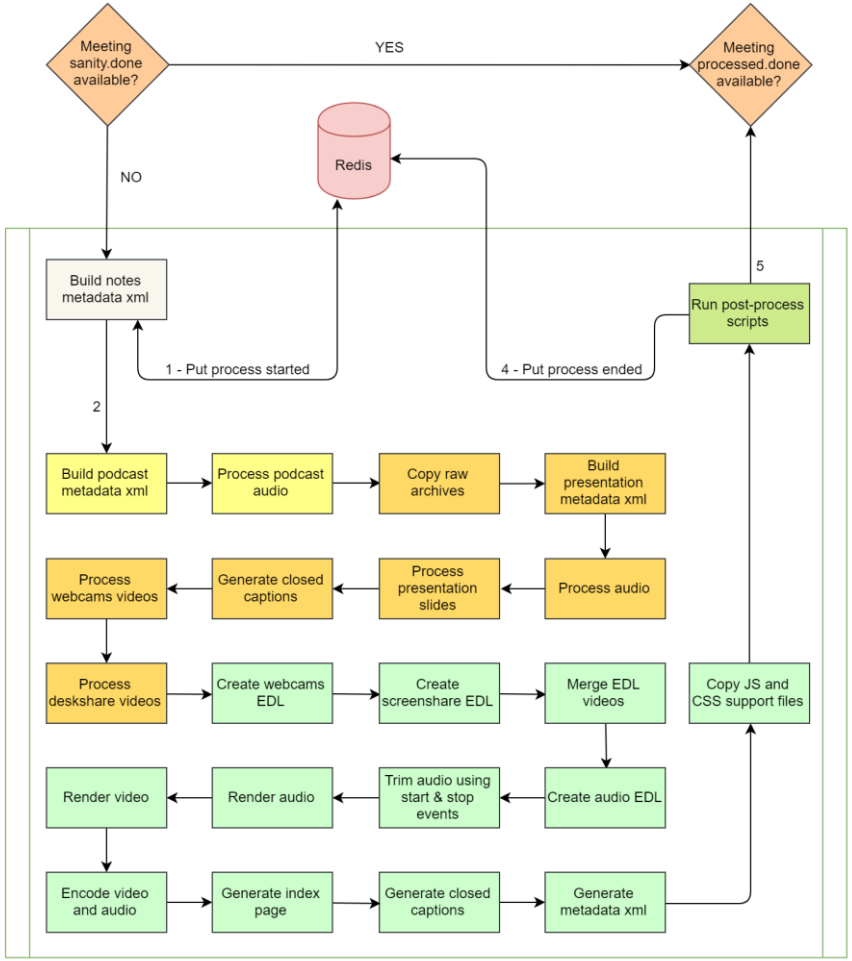

Figure 4. The flowchart of the Process stage of BigBlueButton [8]

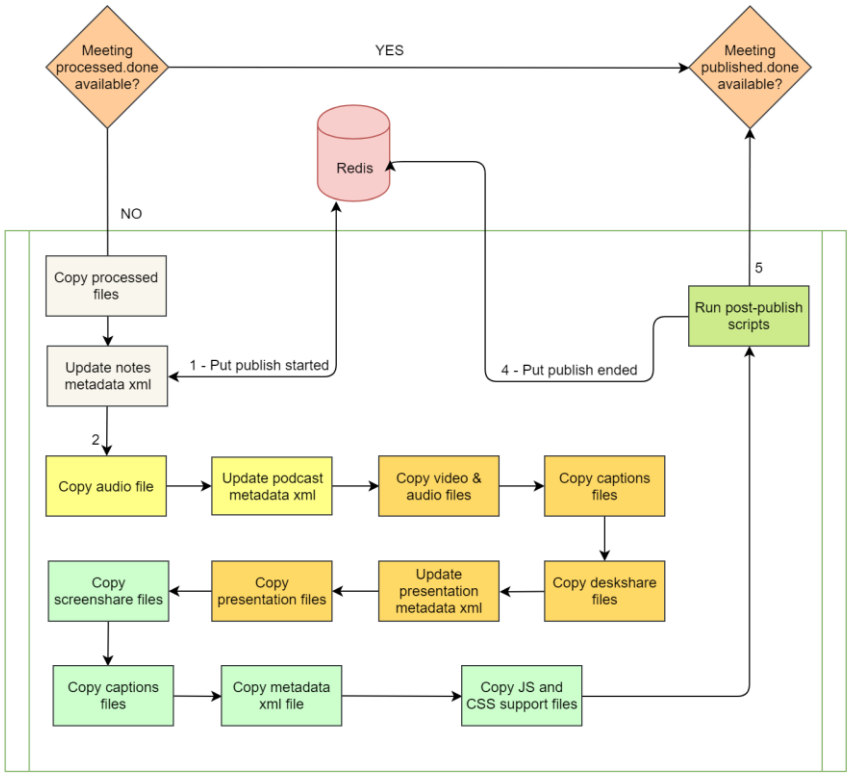

Figure 5. The flowchart of the Publish stage of BigBlueButton [8]

The Sanity stage of the processing pipeline checks the archived files by verifying their validity against a set of quality rules, to ensure that they are fit for processing. Among other things, it examines media files to ensure they are not of zero length and confirms whether events have been archived correctly. The steps performed during the Sanity check stage are described in Figure 3.

The Process stage of the processing pipeline processes archived files according to one or more processing workflows. The default (built-in) workflow is called "presentation", and it generates an HTML5-based recording of the meeting that can be played in any modern web browser. The steps performed during the Process stage are described in Figure 4.

This is the stage of the processing pipeline that our research is focused on, as it is the one requiring the most time, memory, and processing power to complete.

Last, but not least, the Publish phase generates metadata for the recording and transfers the related processed files to a directory that is made publicly available for future playback. The steps performed during the Publish stage are described in Figure 5.

3. Optimization of the audio-video post-processing pipeline

BigBlueButton is an open source project that constantly receives improvements and new features from a large community of software developers [9][10][11][12]. While many of these developers are motivated only by their passion to create highly quality software that helps others, some of them are even financially supported by companies that integrate BigBlueButton into their own products or service offers.

Due to this high interest, some of the software contributions focus on the optimizations of the backend processing engine, which integrates the processing pipeline, and which is implemented in the form of a software component called bbb-record.

Along the time, many ideas for making bbb-record faster have been discussed and some of them even implemented. Among these, we identified the following directions:

- use more processing threads for the video encoding software (called ffmpeg);
- running multiple processioning pipelines to perform simultaneously, to process more meetings at the same time (i.e. instead of queueing them);
- postpone processing when server is less busy (e.g. during nighttime) to deal with the extra CPU load.

However, none of these solutions actually optimizes the processing pipeline, in the sense of making it faster by using less CPU power. We decided to investigate this direction of improving bbb-record service, by changing the audio-video processing flow.

3.1. Existing process stage of the processing pipeline

After a few simple benchmarks, we found out that Archive, Sanity check, and Publish stages take little time because they do mostly file parsing and copying operations. However, our analysis revealed that Process stage requires the most CPU, because it is processing multiple audio and video files with the help of the open-source video processing software ffmpeg. So we focused our research on this part.

Upon reviewing the open-source code for bbb-record-process program, we identified the following media inputs:

- an events list, the contains information about the changes occurred during the meeting (e.g. moderators start/stop recording, presenters start/stop screen-sharing, participants turn on/off their video camera, participants join and leave the meeting, etc),
- an audio file recorded by freeswitch software (using the Opus audio codec),
- zero or more video files from webcams recorded by kurento software (in webm format, using the VP8 video codec and Opus audio codec),
- zero or more video files from screen-sharing sessions recorded by kurento software (in webm format, with VP8 video codec and Opus audio codec),

On these input files, the bbb-record-process performs the following processing steps:

1. use the edl ruby library to cut and merge sections of the audio file according to the recording on/off events and stores the result in recording.flac (encoded using the Flac audio codec)
2. use ffmpeg to encode recording.flac to file audio.encode.ogg (encoded using the Vorbis audio codec)
3. use ffmpeg to encode recording.flac to file audio.encode.webm (encoded again using Vorbis audio codec, with the same settings)
4. use the edl ruby library to cut and merge sections of the webcam video files according to the recording on/off events and stores the result in file webcams.ts (encoded using MPEG-2 video codec)
5. use ffmpeg to combine audio file recording.flac with the video file webcams.ts, and encode the output to:
   a) webcams.mp4 (encoded using H264 video codec and AAC audio codec), and/or
   b) webcams.webm (encoded using VP8 video codec and Opus audio codec)
6. use the edl ruby library to cut and merge sections of the screen-sharing video files according to the recording on/off events and stores the result in deskshare.ts (encoded using MPEG-2 video codec)
7. use ffmpeg to encode deskshare.ts to:
   a) deskshare.mp4 (encoded using H264 video codec), and/or
   b) deskshare.webm (encoded using VP8 codec)

3.2. Optimization of the process stage of the processing pipeline

Once we outlined the entire process, one can quickly notice that many file unneeded conversions and encodings are happening (Flac, Vorbis, Vorbis again, MPEG-2 etc), which needlessly consume CPU and also IO (i.e. the Flac and TS files become quickly quite large).

We changed the processing workflow in order to perform the following steps:

1. use the edl ruby library to cut and merge sections of the audio file according to the recording on/off events and stores the result in recording.ogg (encoded using the Vorbis codec)
2. use ffmpeg to convert recording.ogg to audio.encode.ogg (just copy the audio Vorbis data stream)
3. use ffmpeg to convert recording.ogg to audio.encode.webm (just copy the audio Vorbis data stream between the two different container formats)
4. use the edl ruby library to cut and merge sections of the webcam files according to the recording on/off events and store the result in webcams.mp4 (encoded using H264 video codec and a new ffmpeg settings)
5. use ffmpeg to combine audio file recording.ogg with video webcams.mp4, and output the result to:
   a) webcams.mp4 (copy the video H264 data stream and encode only the audio stream using AAC codec), and/or
   b) webcams.webm (encoded using VP8 video codec with new ffmpeg settings, and copy the audio Vorbis data steam)
6. uses the edl ruby library to cut and merge sections of the screen-sharing files according to the recording on/off events and output the result to deskshare.mp4 (encoded using H264 video codec and a new ffmpeg settings)
7. uses ffmpeg to encode deskshare.mp4 and output the result to:
   a) deskshare.mp4 (copy the video H264 data stream), and/or
   b) deskshare.webm (encoded using VP8 video codec with new ffmpeg settings)

One can quickly notice that we redesigned the entire workflow to reuse encoded data as much as possible (i.e. copy data steams using the “ffmpeg -codec copy”).

We also changed the a few ffmpeg settings in order to encode H264 and VP8 faster:

- use the “superfast” preset

- use constant rate factor (crf) of 28
- use a constant keyframe rate of 1 in 10 seconds
- for VP8 codec (i.e. libvpx-vp9 ffmpeg coded), we also set the "realtime" preset and multi-threading (2 threads)

3.3. Benchmarking the changes

We tested the changes to the Process stage with multiple meetings, with durations varying between 2 and 15 minutes, some of them having only audio streams, and others having also webcam and screen-sharing video streams. To rule out additional influencing factors, all the tests were performed on the same server, with no additional loads.

For each meeting, we compared the processing time of the existing BigBlueButton pipeline (T1) and the processing time of our proposed pipeline (T2), and we identified the following differences reported in Table 1.

Table 1. Relative speed improvements for processing pipeline.

Use case	Relative speed (T2/T1)
Audio-only conferences (i.e. no webcam or screen-sharing streams)	33%
Audio-video conferences (i.e. with webcam and/or screen-sharing streams) with MP4 final files	25%
Audio-video conferences (i.e. with webcam and/or screen-sharing streams) with WebM final files	18%
Audio-video conferences (i.e. with webcam and/or screen-sharing streams) with MP4 and WebM final files	14%

While processing speed is an important factor for this change, retaining high quality of the audio-video recordings is vital, since any change that would reduce the recording quality of the meeting would not be accepted by the BigBlueButton's core maintainers.

Therefore, we performed an additional study on the quality of the video streams (webcams and screen-sharing) created by the proposed pipeline compared to the one created by the current pipeline. For this study, we compared the frames from the original video streams with that of video streams created by the current and proposed processing pipelines for the MP4 output files (using the NCC metric of the ImageMagic's compare tool). The average difference between the original and proposed pipeline was approximately 0,05%, which means that the MP4 output files after these changes are 99.95% similar to those transmitted by the meeting participants.

Another important factor for the processing pipeline is the compression factor, or equivalent, the file size of the output files. We compared size of the output files for the new pipeline against the current pipeline. The results show that MP4 output files are 15 to 30% smaller for the new pipeline, compared to the current one, and 20-30% smaller for WebM files.

For audio files, the quality and size of output files are virtually the same between the two pipeline implementations because the encoding is done with same parameters. However, the new pipeline is faster than the current one (as indicated in Tabel 1).

4. Conclusions

The general objective of our research was to optimize the audio-video processing pipeline of the BigBlueButton video-conferencing software, with the specific objectives to reduce its computational load with minimal loss of content quality and also to reduce the required bandwidth if possible. Based on our research, we identified the stage where these goals can be achieved (i.e. the Process stage), we proposed a new flow for the it, and analyzed the results. An implementation of these changes was

submitted as a pull request to the GitHub repository of the BigBlueButton project, it was reviewed by core developers, and accepted into the main branch of the project.

Acknowledgements: This work was supported by grant no. 383/390059/04.10.2021, project cod ID / Cod MySMIS: 120201: Innovative integrated maritime platform for real-time intervention through simulated disaster risk management assistance in coastal and port areas – PLATMARISC.

References

[1] Kumar, Krishan, et al. "An analysis of Big Blue Button remote teaching tool in an Information Systems undergraduate course." Pacific Asia Conference on Information Systems. Vol. 56. 2021.
[2] Galindo-González, Leticia. "The BigBlueButton in teaching-learning processes, environmental education in ecotecnologies for sustainability." Journal of Teaching and Educational Research 6.17 (2020): 17-29.
[3] Suga, Hiroshi. "A comparison of bandwidth consumption between proprietary web conference services and BigBlueButton, an open source webinar system." Bioresource Science Reports 13 (2021): 1-11.
[4] Turulja, Lejla, Amra Kapo, and Merima Činjarević. "Engage me through BigBlueButton: Student engagement when attending classes online Is the only option." Fostering meaningful learning experiences through student engagement. IGI Global, 2021. 1-15.
[5] Annisa, Ekaputri Wahyudi, and Suroso Bambang. "Students' Perception Towards the Use of Bigbluebutton in English Education Department Universitas Muhammadiyah Purwokerto." Book Chapter Pedagogical Innovations in Education (2021): 114-117.
[6] Raskina, I., et al. "The usage of bigbluebutton for organizing independent work of students in moodle." European Proceedings Of Social And Behavioural Sciences EPSBS. 2021.
[7] Sánchez Godínez, Elisa, and María Stella Stradi Granados. "Learning Innovation through BigBlueButton technological tool implementation in statistics courses at UNED, Costa Rica." Revista Innovaciones Educativas 22.33 (2020): 106-123.
[8] ***, https://docs.bigbluebutton.org/dev/recording.html
[9] Geislinger, Robert, et al. "Live Subtitling for BigBlueButton with Open-Source Software." Interspeech. 2021.
[10] Ukoha, Chukwuma. "As simple as pressing a button? A review of the literature on BigBlueButton." Procedia Computer Science 197 (2022): 503-511.
[11] Uhl, Christian, and Bernd Freisleben. "Performance Improvements Of Bigbluebutton For Distance Teaching." IADIS International Journal on Computer Science & Information Systems 17.1 (2022).
[12] Rocha, Daniel Petri. "Development of a Sophisticated Session Recording Exporter for the BigBlueButton Web Conferencing System." (2021).