A method to increase video streaming speed and efficiency of storage video using multithreading and Random Access Memory

Doan Van Hoa[1], Thai Trung Kien[1], Hoang Dinh Thang[1],
Dao Khac Huan[1], Nguyen Xuan Bac[1*], Hoa Tat Thang[2]

[1]Institute of Information Technology, Academy of Military Science and Technology, Cau Giay, Hanoi, Vietnam;
[2]Institute of Information and Communication Technology, Military Technical Academy, Cau Giay, Hanoi, Vietnam.
[*]Corresponding author: xuanbac93bn@gmail.com
Received 8 Nov. 2023; Revised 28 Dec. 2023; Accepted 5 Jan. 2024; Published 25 Feb. 2024.
DOI: https://doi.org/10.54939/1859-1043.j.mst.93.2024.171-174

ABSTRACT

There are numerous vital applications for online video storage and network transmission today. With exceptional scalability and browser and device compatibility, Hypertext Transfer Protocol (HTTP) live streaming (HLS) is today's most popular video and audio transmission protocol. In the article, the authors proposed a method of using multithreading and Random Access Memory (RAM) to store and transmit videos to help the server achieve higher speed, improve video transmission speed and video storage efficiency. The results of this research will help multimedia server developers evaluate the appropriate balance of resource usage while still ensuring the quality of online video transmission with a large number of videos and viewers.

Keywords: Multithreading; Streaming; RAM; Storage.

1. INTRODUCTION

The main goal of building a media server is to make it possible for users to watch online videos smoothly and professionally. The process of streaming video from the media server to the user is regulated by transport protocols, the most prominent of which are Real Time Streaming Protocol (RTSP), Real-Time Messaging Protocol (RTMP), HLS, and (Web Real-Time Communication) WebRTC.

Video streaming protocols deliver data over the Internet whenever watching a live stream or video on demand. These can sit in the application, presentation, and session layers in the OSI 7-layer model [1]. HLS video transmission requires building a media server. The media server will receive the data directly from the camera, then analyze it, break it down and save it to TS files. When a user requests to watch a video, the server sends those TS files to the user by HTTP. HLS breaks the video files into downloadable small HTTP files and passes them using the HTTP protocol. Clients' devices load these HTTP files and play them back as video [2].

In the article [3, 4], the author has studied the influence of the Internet, and joint problems in the video transmission process, such as delay, transmission error, bandwidth, and how they affect the quality of streaming video through the Internet over HTTP and Transmission Control Protocol (TCP) and the performance of the media server. The solution that this article poses is a user-side player algorithm, which optimizes the player's buffer without mentioning the media server's optimal performance.

The closest to our research is the article [5], in which the author also set up a media server to process the video stream from Internet Protocol (IP) Camera and stream it to web users by HLS protocol. However, the author has not tested with many cameras and has not evaluated and given a method to increase the processing speed of the media server.

When multiple cameras connect and send data to the server simultaneously, the server can use one thread for sequential processing of video streams or numerous threads for concurrent processing [6]. After being analyzed by the server, the video streams will be saved as TS files; these files can be saved directly on RAM [7] or held on the computer's hard drive.

Each method has different advantages and disadvantages. This paper focuses on empirical analysis to determine each technique's pros and cons. Therefore, the authors propose a solution for the server to use multithreading to process data, and the TS files will be saved on the server's RAM. Although using many resources, this method will help the server operate stably and perform better.

2. PROPOSED METHOD

In this study, we simulated from 1 to 200 cameras that send data to the HLS media server and recorded the resource usage and performance of the media server in the case of saving the ts file on the memory drive stored in the main memory. The histogram in the cases has been made.

For research purposes, a media server has been built that helps to receive video data from the camera and store and transmit the video to the user by HLS protocol. The entire process of storage and transmission is depicted in *figure 1*.

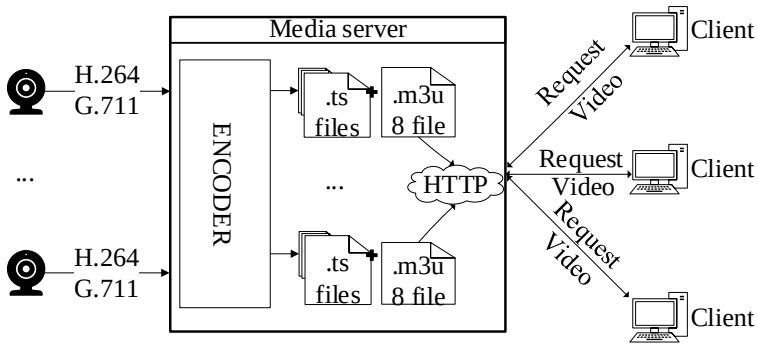

Figure 1. *Video transmission by HLS protocol.*

The data received from the camera will be analyzed by the server, packed into TS files, and transmitted to the viewer by HTTP. The authors have also built a website where the videos sent to viewers from the media server can be viewed directly.

To evaluate the performance of the media server, the authors simulated several 1 to 200 camera sources, sending H.264 video data and G.711 audio to the server simultaneously. The server will have to arrange the threads for packing and split the image frames into TS files accordingly. Then, transfer the TS files to the correct web address where the user is requesting to watch the video. The server uses multiple threads to process incoming video sources and TS files stored directly in RAM (*figure 2*).

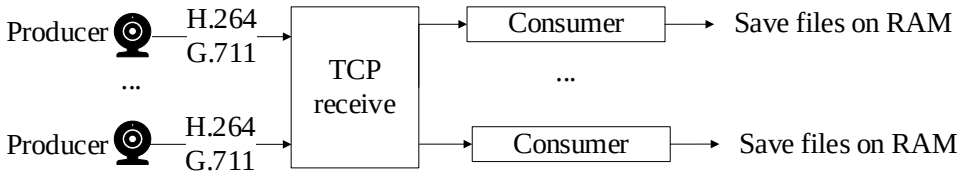

Figure 2. *The server uses multiple threads, files are stored in RAM.*

The results of the tests are presented in section 4. The authors evaluate the methods for efficiency (delay) and resource usage.

Regarding delay: The time between the camera sending the image down and the server receiving it until it saves the image in the TS file. Note that the next part is to transfer the ts file to the client by HTTP; the client receives and uses a particular player to cache and play. The latter part deals with the speed and stability of the network, and the way players handle buffers has nothing to do with server performance, so the authors will not study it in this article.

Regarding resource usage: The amount of RAM the server uses in the cases.

3. RESULTS AND DISCUSSION

Four cases have been done as follows:

- Case 1: The server uses a single thread, all TS files, m3u8 files are stored on the hard drive. According to this method, the video sources will be processed, extracted, and saved into TS files directly on the server's hard drive.

Advantages: Simple and easy to implement, consumes fewer machine resources. The disadvantage is that when many devices are sent simultaneously, the server will not be able to handle it in time, causing the unprocessed video data to accumulate, leading to loss of frame and memory leaks. As a result, in *figure 3a*, when more than 10 cameras are connected to the server, the server cannot handle it, leading to a very high delay (> 100 s). This method is only suitable when dealing with very few devices.

- Case 2: Server uses a thread, TS files are stored in RAM. As in case 1, the server is only used to process, extract, and pack video data for all cameras, but the files are stored in RAM instead of on the hard drive. This change increases processing speed and reduces packet loss. As seen in *figure 3b*, when the number of devices is up to 200, the video delay is still stable at 0.02 s. However, the RAM resource that the server uses is quite large, in this case, about 3 GB.

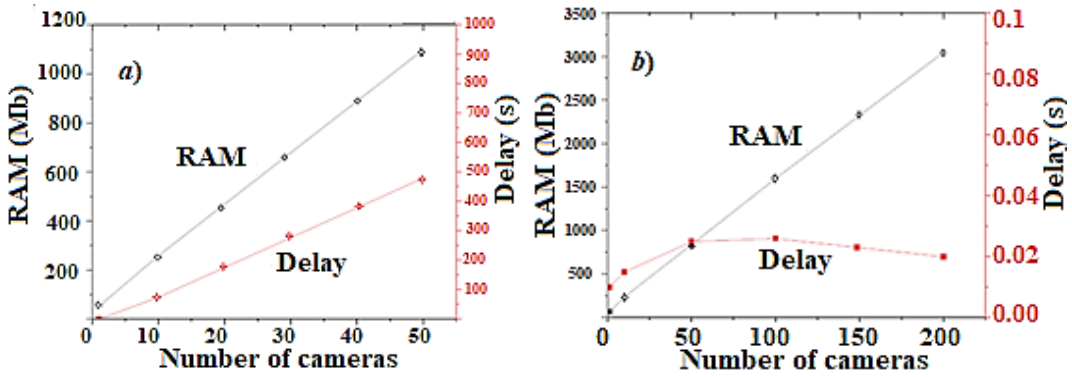

Figure 3. *Method using 1 thread, saving files on the hard drive (a) and saving files on RAM (b).*

- Case 3: The server uses many threads, and the files are stored on the hard drive. Now, we will try to use the multi-thread method, and each camera source is taken care of by one thread, avoiding the situation of affecting each other. This case will save the TS files on the hard drive.

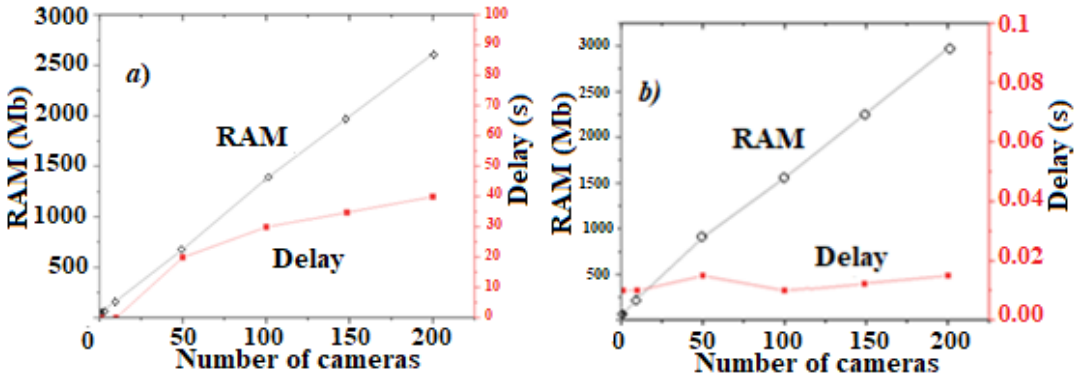

Figure 4. *Method using multiple threads, saving files on the hard drive (a) and saving files on RAM (b).*

According to *figure 4a*, using multiple threads to extract and store data from cameras has avoided data congestion when many cameras send data simultaneously. However, the slow processing speed makes the delay significant.

- Case 4: The server uses many threads, and files are stored in RAM. We will use two methods to help the server achieve the highest speed in this case. For each connected camera, there will be a thread that handles the data processing of that camera, and the data will be saved as TS files directly on RAM. This method helps the server's processing achieve high results; even when many cameras are also connected, the delay is still stable at 0.015 s. However, the amount of RAM used is relatively large, about > 3 GB (*figure 4b*).

4. CONCLUSIONS

Live video streaming is an indispensable part of today's technology era. HTML5 solutions - transmitting and watching videos over the web using HLS streaming method is the solution with outstanding advantages of scalability and compatibility with many browsers; this is also the most popular video streaming protocol. In implementing the solution of live video transmission using HLS, it is necessary to have a stable and high-performance media server.

The results show that using and saving TS files on a disk is only suitable for a small number of video sources. If the number of video sources is significant, frame loss and memory leaks may occur. The method of using multiple threads and storing them in memory can solve the above problem with a large number of cameras connected to the server. However, this method requires the server to have a robust configuration with much RAM.

Acknowledgment: *The authors would like to thank the financial support of the science and technology project at the grassroots level of Institute of Information Technology, Academy of Military Science and Technology.*

REFERENCES

[1]. Traci Ruether, "*Streaming Protocols: Everything You Need to Know*", (2022). [Online]. Available: https://www.wowza.com/blog/streaming-protocols

[2]. *"What is HTTP Live Streaming"*. (2021). [Online]. Available: https://www.cloudflare.com/learning/video/what-is-http-live-streaming/

[3]. Biernacki, Arkadiusz, and Kurt Tutschku. "*Performance of HTTP video streaming under different network conditions.*" Multimedia Tools and Applications 72.2: 1143-1166, (2014).

[4]. Yan, He, Huang Haocheng. "*Performance Measurement and Bottleneck Analysis for Streaming Media Servers.*" 3rd International Conference on Multimedia Technology (ICMT-13). Atlantis Press, (2013).

[5]. Yang, Gil Jin, Byoung Wook Choi, and Jong Hun Kim. "*Implementation of HTTP live streaming for an IP camera using an open source multimedia converter.*" International journal of software engineering and its applications 8.6: 39-50, (2014).

[6]. Bill Wagner, Poojapoojari, Pkulikov, "*Using threads and threading*", (2022). [Online]. Available: https://docs.microsoft.com/en-us/dotnet/standard/threading/using-threads-and-threading

[7]. *"Memory Stream",* (2021). [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.io.Memorystream?view=net-6.0.

TÓM TẮT

Phương pháp tăng tốc độ truyền và hiệu quả lưu trữ video sử dụng xử lý đa luồng và bộ nhớ RAM

Lưu trữ, truyền tải video trực tuyến qua mạng có rất nhiều ứng dụng quan trọng trong giai đoạn hiện nay. Với những ưu điểm vượt trội về khả năng mở rộng, khả năng tương thích với nhiều loại thiết bị và trình duyệt thì HTTP live streaming (HLS) đang là giao thức truyền video, âm thanh phổ biến nhất hiện nay. Trong bài báo nhóm tác giả đã đề xuất phương pháp sử dụng đa luồng và bộ nhớ RAM để lưu trữ và truyền tải video giúp máy chủ đạt tốc độ cao hơn, nâng cao tốc độ truyền tải và hiệu quả lưu trữ video. Kết quả của nghiên cứu này sẽ giúp nhà phát triển máy chủ đa phương tiện đánh giá, cân đối việc sử dụng tài nguyên một cách hợp lý, mà vẫn đảm bảo chất lượng truyền tải video trực tuyến với số lượng lớn nguồn video và người xem.

Từ khoá: Đa luồng; Truyền video; Bộ nhớ RAM; Lưu trữ.