An efficient scheduling multimedia transcoding method for DASH streaming in cloud environment

Linh Van Ma, Jaehyung Park, Jiseung Nam, Jonghyun Jang & Jinsul Kim

Cluster Computing
The Journal of Networks, Software Tools and Applications

ISSN 1386-7857

Cluster Comput
DOI 10.1007/s10586-017-1259-8

Springer

Cluster Comput
DOI 10.1007/s10586-017-1259-8

An efficient scheduling multimedia transcoding method for DASH streaming in cloud environment

Linh Van Ma[1] · Jaehyung Park[1] · Jiseung Nam[1] · Jonghyun Jang[2] · Jinsul Kim[1]

Received: 3 May 2017 / Revised: 23 September 2017 / Accepted: 9 October 2017

Abstract As a result of technological evolution, streaming service providers have been dealing with the problem of delivery multimedia content to the diversity of devices with different resolutions. This issue can be solved by using dynamic adaptive streaming over hypertext (DASH) transfer protocol. However, a transcoding job in DASH requires a lot of computation resource which could lead to delaying the starting of multimedia streaming. Recently, new studies have addressed novel scheduling methods on video transcoding, but those research did not solve the problem entirely, such as the solution did not concern server performance or speed connection between a server and its requested users. Moreover, the load and speed connection status of the data servers is often unstable, leading to increasing the starting delay. So in this article, we solve such problem by modeling transcoding jobs in the form of an optimization problem and propose an algorithm to find an optimal schedule to transcode video source files. In which, we use moving average method to find average points for a short period to deal with server state changes. In the experiment, we implement our proposed method with DASH to demonstrate the effectiveness of the optimization scheduling method. In the system, we create several servers running on the Docker platform to simulate a cloud environment. Experimental results show that our methodology reduces the time of the transcoding process up to 30% compared to existing research.

Keywords Cloud computing · Adaptive streaming · Transcoding · Data replication · Docker

✉ Jinsul Kim
jsworld@chonnam.ac.kr

Linh Van Ma
linh.mavan@gmail.com

Jaehyung Park
hyeoung@chonnam.ac.kr

Jiseung Nam
jsnam@jnu.ac.kr

Jonghyun Jang
jangjh@etri.re.kr

1 School of Electronics and Computer Engineering, Chonnam National University, 77, Yongbong-ro, Buk-gu, Gwangju 500-757, Korea

2 Electronics and Telecommunications Research Institute, 218 Gajeong-ro, Yuseong-gu, Daejeon 34129, Korea

1 Introduction

The rapid development of today's technology provides a way to access customized services based on demand. For example, some people may be willing to spend money to invest in expensive devices, such as iPhone 7, Samsung Galaxy S7. On the one hand, these devices require the proportion of devices to quality services, such as the S7 requires high-definition (HD) video streaming. On the another hand, people in remote areas do not have much interest in technology. For example, they have quite simple needs such as watching video streaming at low quality on low-profile phones. Therefore, a streaming service which can provide a broad range of video quality from low to ultra-high quality is needed. Furthermore, the number of Internet users is on the rise, and it has reached more than three billion users by 2015 [1]. Consequently, streaming services cannot meet the needs of such large numbers of users with a small number of servers. Thus, cloud computing [2,3] has emerged as a method of sharing resources, data between computers and electronic devices based on user requirements.

Published online: 23 October 2017

With the active development of social services such as Facebook, Twitter, video streaming is necessary to convey information to users because humans are familiar with images rather than sound and words. As mentioned above, we cannot meet the heterogeneous Internet users' video streaming requirements if we do not have efficient streaming delivery solutions. Adaptive streaming [4] has emerged as a technology that can adapt to the uncertainty of network. Specifically, it is a video streaming technology where video quality can vary depending on the state of the network and device performance. A streaming server segments a multimedia file into segments with a duration about 2–10 s. Then, the server creates different versions of those files with different qualities using the same content of multimedia files. Recent studies [5,6] addressed that, we could install the adaptive streaming systems in a cloud environment, which could provide video streaming seamlessly in any network environment as well as improve the system user-friendliness.

Besides the advantages of dynamic adaptive streaming over hyper text transfer protocol, HTTP (DASH), some of the DASH problems on cloud computing are not fully resolved yet, such as the trade-off between files replication and media transcoding in data centers. In a more specific way, when a user uploads data to a server, a central server is responsible for managing the data servers. It decides whether to transcode the uploaded files or replicate the files from other data servers which contain the content files. If the server handles the transcoding file and does not have much computation resource, it may delay the starting of video streaming. Instead of this, the center server might consider replicating DASH files from other servers. Furthermore, recent studies have not focused on the choice of data centers to serve user requests. They only interested in common parameters when choosing a serving server such as bandwidth or server load, not a combination of both. Furthermore, those studies also only consider the present value of cloud measurement parameters such as server load, network throughput. This selection-based method will sometimes result in rough estimates or non-optimal options which lead to increasing the starting delay of a video streaming. Therefore, in this article, we propose a method to reduce the delay which improves the quality of video streaming in DASH. Specifically, by using a server that manages other data servers in the cloud, we manage servers in the cloud and propose a transcoding schedule to optimize the performance of the network as well as reduce the starting delay of DASH streaming. In the termination, this research has two contributions as the following. First, we state recent studies of transcoding and related topics. Then, we propose a heuristic to schedule to reduce transcoding time using several measurement metrics for evaluating the performance of a server.

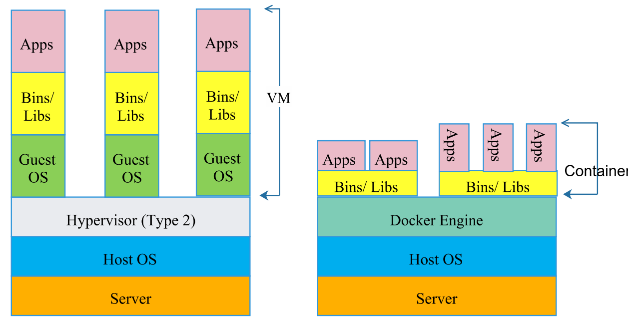

Fig. 1 A comparison between VMs and containers

The rest of paper is organized as follows. In Sect. 2, we present an overview of Docker, DASH streaming in cloud computing and transcoding multimedia in the cloud. We present related works in Sect. 3. In Sect. 4, we introduce the problem of DASH streaming and model this as an optimization problem. In Sect. 5, we describe our experiment and discuss the results achieved. Section 6 presents our findings with future research directions.

2 Background information

2.1 An overview of Docker and virtualization technologies

Docker container is a virtualization technology. The container puts everything, such as system libraries, code, system tools, into a file-system. Especially, we can use the file-system anywhere and execute the system anytime on any operating systems (OSs) supported Docker. The system has the same behavior regardless of OS because it has the same libraries, code, etc. On Linux and Windows system, we have an additional abstraction layer and automation virtualization OS level. Furthermore, Docker containers run independently within a single Linux instance because it uses isolation features of Linux kernel such as kernel namespaces and cgroups. Therefore, it allows a container avoid the overhead of maintaining and starting virtual machines (VMs). Figure 1 shows a comparison between VMs and Docker containers. While VMs perform efficiently at isolation with a complete system, containers work at the process level of OS, which offers them much beneficial of software delivery and system deployment. As shown in Fig. 2, Docker implements a high-level application program interface provided by the Linux kernel, such as namespaces and primarily cgroups, to bring light-weight containers that run processes in isolation.

Linux containers (LXC) [7] is an OS-level virtualization method for running multiple isolated Linux systems (containers) on a control host using a single Linux kernel. A

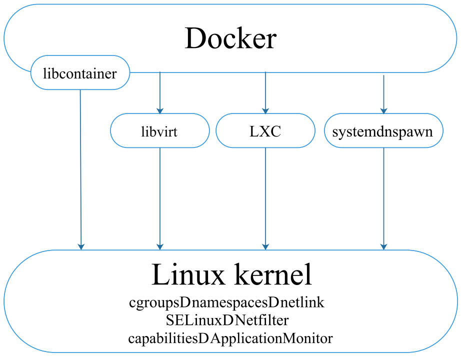

Fig. 2 Accessing virtualization features of the Linux kernel of Docker

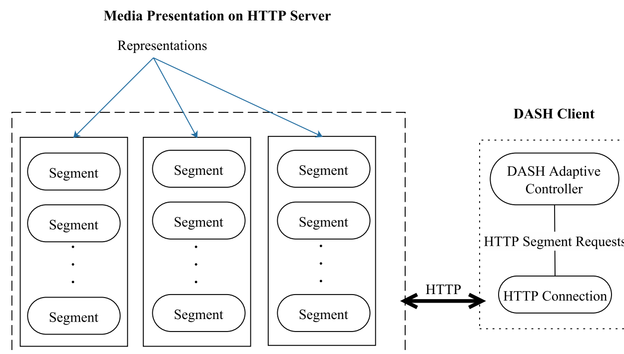

Fig. 3 An overview of dynamic adaptive streaming over HTTP

single server or VM can run several containers simultaneously because Docker containers are lightweight. An analysis in [8] found that a typical Docker user case involves running five containers per host, but that many organizations run ten or more. By using containers, we can provision processes, restrict services and isolate resources to have an almost entirely private view of the OS with network interfaces, file system structure and their process identifier space. In addition, multiple containers share the same kernel, but each container can be constrained to only use a defined amount of resources, such as input/output, memory and central processing unit (CPU).

2.2 Adaptive streaming in cloud environment

Adaptive bitrate streaming (ABS) is a streaming technology which enables high-quality multimedia streaming over HTTP. It changes video streaming representations along with network conditions such as throughput. It makes multimedia content available at a variety of different bitrates so that a client can retrieve the multimedia streaming without noticeable change of video quality or re-buffering while playing back. Also, ABS divides contents into small HTTP-based segments so that a client can obtain a file by using GET or POST command in HTTP. Each segment has a length around 2–10 s. We have several version of ABS such as DASH supported by Moving Picture Experts Group (MPEG), HTTP live streaming supported by Apple and it is streaming standard in iPhone and iPad, smooth streaming by Microsoft, Adobe HTTP dynamic streaming by Adobe. In this research, we only consider DASH system. As shown in in Fig. 3, a DASH client has an adaptive controller which supports the client automatically select video quality based on network conditions. A media presentation description (MPD) in an extensible markup language file managing information of segments as well as video representations. At the initial connection, the client obtains the MPD file to prepare an adaptive strategy to deal with network fluctuations.

Network communication always deals with the problem of optimizing transmitted data over the limited bandwidth. Especially, in mobile network communication, where the maximum transmitted is lower than the wire connection. Intuitively, common sense is to reduce data transmission while maintaining the transmission time and the amount of information over that period is using data compression techniques such as video coding. In fact, video coding compresses data by comparing different parts of a frame with other frames to find redundant information in the frame. Consequently, it uses less information to describe the frame rather than using the original data. Some advanced video coding (AVC) is H.261 [9], AVC/H.264 [10], and high-efficiency video coding (HEVC) [9], which is one of the most efficiency video compression standards in the multimedia, has been using in 4K (resolution of at least 3840 $\times$ 2160) or ultra video streaming projects recently. In HEVC, intra-frame is the based frame where original data of the original pictures locates, which uses to predict information for other frames. An unusual number of a broadcasting system has released their first ultra-HD (UHD) television over digital terrestrial television networks such as Japan, Republic of Korea, France, and Spain. Most of them used the most advantage of compression data technique HEVC. It implies that compression techniques become an essential technology to optimize transmission data over the network.

In a context of HD video transmission and distribution [11], an uncompressed HD digital video is used to demonstrate that a high-quality multi-party video conference based on a transmission of uncompressed HD streams is already achievable. Moreover, the contribution of a hardware architecture for H.264/AVC decoders proposed in [12] showed that the processing capability of the proposed architecture is to support (2048 $\times$ 1024, 30 fps) videos at 120 MHz. In which, a hybrid task pipelining scheme is presented to reduce the internal memory size and bandwidth significantly. Besides, they also proposed an appropriate degree of parallelism for each pipeline task.

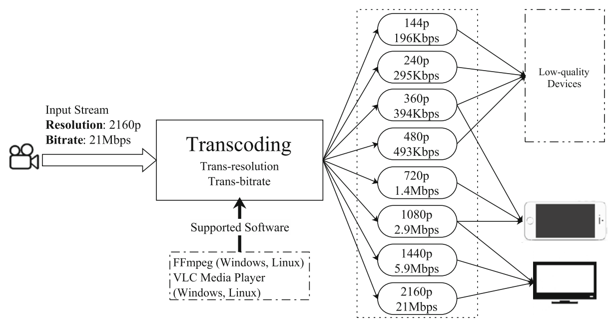

Fig. 4 A process of transcoding a source video to different representations

Nowadays, we are evidencing of the increasing demand for HD multimedia contents in the mobile environment. The increase requires an innovative network design which should satisfy quality-of-service (QoS) requirements of multimedia applications. In the context of HD multimedia transmission, the scalable extension of H.265/HEVC and scalable HEVC (SHVC) [13] are a potential video coding which supports encoding multimedia up to 8K UHD. Furthermore, SHVC deployment reduces implementation cost significantly. It also supports a rich set of scalability features. However, it requires much computing resource to encode and decode multimedia streaming. This intensive computation is a difficult task for mobile devices as they usually have limited computational resource.

2.3 Multimedia transcoding in the cloud

In ABS, transcoding a source video file into different representations is required in the process of making the adaptive streaming as shown in Fig. 4. First, this job reads an input multimedia file and determines the maximum resolution of the input. Then, it transcodes the file into different resolutions which cannot greater than the determined maximum value. As addressed in Sect. 2.2 and [14], transcoding time and computation resource requirements for making DASH files (i.e., transformatting .mp4 files to .m4s files), are only a few milliseconds. Therefore, in this study, we focus on scheduling methods for the video transcoding jobs which require intensive computation resources. Intuitively, low resolution and short video duration do not consume a lot of resources compared to high resolution and long duration one. Besides, video transcoding time increases exponentially when the number of transcoded resolutions and video length increase. Therefore, estimating transcoding time could lead to the reduction of time for an uploaded video.

As addressed in research [15], the authors divided multimedia transcoding into three categories which are online, offline and hybrid transcoding. First, they referred the offline one as performing transcoding before the delivery process. In contrast, the online one can be seen as a real-time transcoding task. It only performs if a client requests a video segment with a particular quality resolution. Therefore, the task comes along with the process of video delivery and requires predictive algorithms which can deduce the right time to start transcoding tasks.

3 Related works

Cloud computing is now a leading-edge Internet technology, in which virtualization technology is one of the important parts of cloud computing system. Virtualization in computing refers to the act of creating a virtual (rather than actual) version of something, including but not limited to a virtual computer hardware platform, OS, storage device, or computer network resources. Recently, Docker is a new type of virtualization technology which was addressed in [16] by describing Docker applications and its advantages in the cloud computing.

High latency, network congestion, and network bottlenecks are some of the problems in cloud computing. By moving from centralized to a decentralized paradigm, edge computing could offload the processing to the edge which indirectly reduces application response time and improves overall user experience. An evaluated Docker [17], a container-based technology as a platform for edge computing presented with four fundamental criteria as the following: (1) deployment and termination, (2) resource and service management, (3) fault tolerance and (4) caching. Their evaluation and experiment showed that Docker provided fast deployment, small footprint and good performance which make it potentially a viable edge computing platform.

Current Docker-based container deployment solutions are aimed at managing containers in a single-site, which limits their capabilities. As more users look to adopt Docker containers in dynamic, heterogeneous environments, the ability to deploy and efficiently manage containers across multiple clouds and data centers becomes of utmost importance. Furthermore, the authors [18] proposed a prototype framework, called C-Ports, that enables the deployment and management of Docker containers across multiple hybrid clouds and traditional clusters while taking into consideration user and resource provider objectives and constraints. The framework leverages a constraint programming model to allocate or deallocate resources as well as to deploy containers on top of these resources. Besides, the author [8] examined how the popular emerging technology Docker combines several areas from systems research—such as OS virtualization, cross-platform portability, modular re-usable elements, and versioning, to address the challenges of computational reproducibility.

Cloud computing provides a variety of services with the growth of their offerings. Due to efficient services, it faces numerous challenges. Virtualization offers users a plethora computing resources by Internet without managing any infrastructure of VM. With network virtualization, VM manager gives isolation among different VMs. But, sometimes the levels of abstraction involved in virtualization have been reducing cloud workload performance which is also a concern when implementing virtualization to the cloud computing domain. Consequently, the authors in [19] explored how the vendors in the cloud environment are using containers for hosting their applications and the performance of VM deployments. In addition, they also compared VM and LXC on the QoS, network performance, and security evaluation.

In adaptive streaming, transcoding multimedia to different bitrates and qualities requires extensive computation resource. Specifically, it takes most of the time in the stages of delivering video streaming to users. Therefore, we need a cloud environment that can handle such high computing demands. Similar to our approach to reducing the time of transcoding, the authors in [14] proposed a video transcoding scheduling method for DASH in the cloud. First, they prioritized each job of transcoding and adjusted transcoding mode based on the cloud system load. In that way, they reduced video transcoding completion time and balanced the cloud system load as well as smoothing video playback at a client side. Furthermore, the authors in [6] formulated an optimization problem in multimedia on-demand. In the problem, they minimized the total operation cost of delivering resources based on a tradeoff between bandwidth, caching and transcoding costs. Besides, they also found an optimal strategy to deliver DASH segments for users. As a result, their approach significantly saved costs compared to existing methods in multimedia streaming.

In the approach of optimization problems, the authors [20] presented an integer linear program to maximize user quality experience. They also proposed a heuristic algorithm that scaled to a large number of videos and users in the cloud. As a result, their system performance led to the optimal global solution compared to the current industry standards which could lead the optimization problem to sub-optimal solution.

4 Overview of proposed cloud streaming system

In DASH streaming, when a client uploads a multimedia file to the DASH server, transcoding software is responsible for making copies of files with different qualities. This job requires a lot of computation resources as well as time to complete the process of transcoding. It is hard for servers with a heavy load to handle the transcoding in real time. Moreover, a client can upload more than one file. Especially, the server load capacity is heavier as it has to handle more requests. Therefore, the management of DASH streaming server is necessary to improve the performance of the DASH streaming system. As shown in Fig. 5, the management server does two important tasks. First, it serves as a normal DASH streaming server. Secondly, it has the ability to manage other servers so that it can redirect requests from the client to other servers. The management server has no additional responsibility to a request from a client when it redirects the request to other servers. To do so, the management server requires high computation resource to accomplish such tasks, and its network speed is sufficient to provide real-time response to a significant number of requests from clients.

In this article, we use seven parameters to evaluate the performance of a DASH streaming server. If we manage server performance efficiently, we can know how quickly a server responds to a certain request. Regarding local measurement, we have; (1) free memory, (2) working CPU, (3) load average, (4) total memory. These local primary metrics represent the current status of a system server. Regarding network measurement, we have; (5) throughput upload, (6) ping, (7) throughput download. Formally, we can name these parameters respectively as the following: m_1, m_2, m_3, m_4, m_5, m_6, m_7. As shown in Fig. 6, management server periodically receives measurement information from DASH servers. Then, it finds a server that is best serving a coming request from a client. Besides, it orders the DASH servers into an order. Furthermore, the management server also shares information about the order for all other servers through a synchronization mechanism using a file-sharing protocol such as file transfer protocol (FTP). Thus, DASH servers use this information for the purpose of retrieving DASH files from other servers in case it does not serve a transcoding request from a client. This technique is a solution for sharing DASH files of a server when it contains a multimedia video while other servers do not have the video content. In addition, a server can retrieve DASH files from other servers based on metric information from the available parameters. For example, Servers A and B have the same DASH video, and Server A is faster than B 25%. If Server C does have the video content, it can get 75% video DASH files from A and 25% DASH files from B. We can use any file-sharing protocols to get these DASH files such as FTP.

Before the modeling paradigm, we introduce the moving average technique to reduce the variance of measurement values of a server such as a load, throughput. For example, a server performs requests coming from clients continuously. Intuitively, the values that we measure at a given time do not accurately describe the state of the server. To reduce the variance of values, we use a moving average to calculate the mean of server parameters over a given period.

In statistics, moving average is a calculation to analyze data sets. Especially, it is commonly used to analyze time series to smooth out short-term fluctuations and highlight

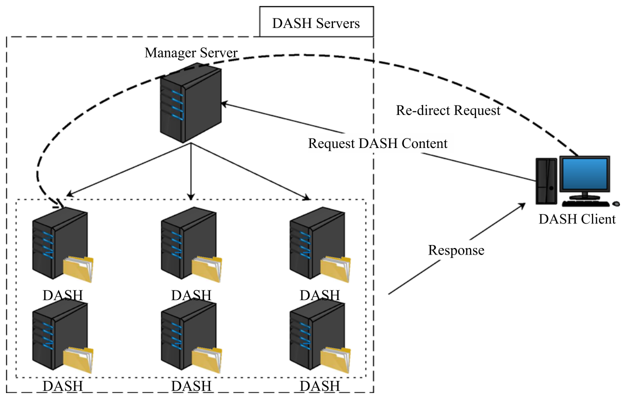

Fig. 5 DASH cloud streaming system overview

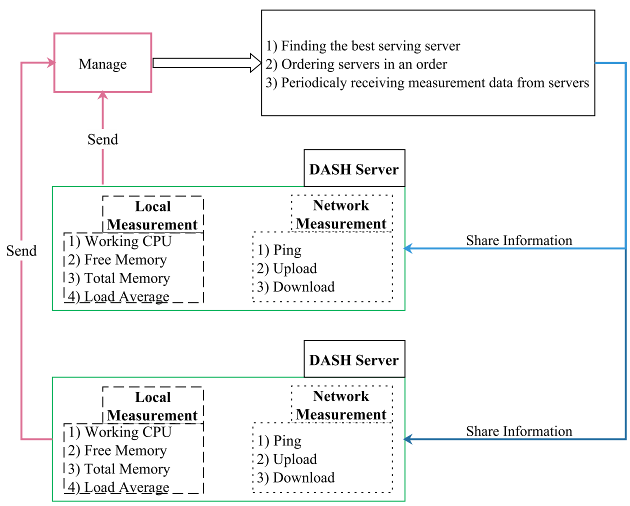

Fig. 6 Management server manages parameter from other servers

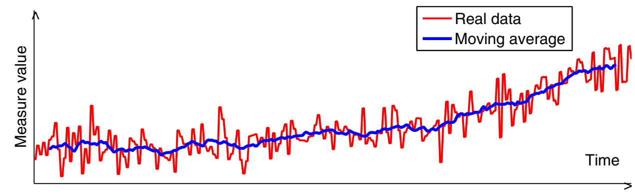

Fig. 7 Moving average calculation

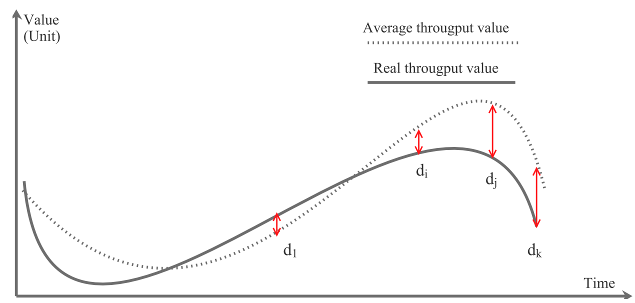

Fig. 8 Reducing fluctuation of data using moving average

longer-term trends or cycles as shown in Fig. 7. It works by creating a series of average points of the data set. More specifically, the first element of the moving average is calculated by averaging the initial set of data. Then, the next element is obtained by shifting forward that excludes the first number of the series and includes the next number following the original data set. Continuously, the process creates a new data set, then averages and repeats over the data set. There have several forms of the moving average: simple average, and cumulative average, or weighted average forms. In this research, we use cumulative moving average with a given number of measurement points in which data arrive in order as streaming data.

Values in the near future are the result of changes in the past. In other words, the change of past parameter values affects the values in the near future. It's an idea to predict the average value of DASH streaming server parameters. As shown in Fig. 8, we compute the mean of the values for a given period of time with approximately k steps. Each step is the value at the time we perform the estimation process to retrieve the current server parameters. At each step, we perform a subtraction between the real and average values, which results in the average value of k steps. We anticipate the average value V_i of any parameter v_i of the next step by using (1) ($i \in \mathbb{N}$).

$$V_{n+1} = V_n + \sum_{i=n-k}^{n} \frac{v_i - V_i}{k}. \quad (1)$$

As mentioned above, we can assign weights to each parameter $m_1, m_2, m_3, m_4, m_5, m_6$ and m_7 respectively by the following $w_1, w_2, w_3, w_4, w_5, w_6$ and w_7 and satisfies (2), to sort the servers in an order. The total metric measurement of a server can be given by (3). Each server has a different total metric, we order the servers into an order by using the total metric.

$$\sum_{i=1}^{7} w_i = 1, \quad (2)$$

$$M = \sum_{i=1}^{7} m_i w_i, \quad (3)$$

$$VTT = \gamma N_r^a D_v^b. \quad (4)$$

When receiving a request from a client, a server can decide whether to get a source video file to transcode, or retrieve all existing DASH files. This method depends on two factors which are local and network measurement. If the server has not sufficient computation resource for transcoding a multimedia file in a short time, it replicates data from other servers. For example, network connection throughput between a server and a client is fast enough to meet the ability of response an incoming request from clients in a short time, could lead to improving DASH cloud performance. On the basic of the research [14], we estimate transcoding time VTT for a video by using (4) where N_r represents number of transcoding resolutions, D_v describes the video length, $\gamma \in [0, 1]$ is a effect factor in the estimation. a, b are the fitting coefficients which can be achieved from [14] as shown in (5).

$$[a\,b] = \begin{cases} [0.796 \quad 0.621] \quad (low) \\ [1.152 \quad 0.700] \quad (medium) \end{cases} \tag{5}$$

$$\begin{aligned} \min_{S_{lj}} \quad & \sum_{j} S_{lj} \\ \text{subject to} \quad & \sum_{j=1}^{n}\sum_{i=1}^{4} m_{ij} \le \sum_{j} L_j \\ & \sum_{j=1}^{n}\sum_{i=5}^{7} m_{ij} \le \sum_{j} N_j \\ & S_{lj} = \alpha \sum_{i=1}^{4}(-1)^i m_{ij} w_{ij} \\ & \quad + \beta \sum_{i=5}^{7}(-1)^i m_{ij} w_{ij} \\ & \forall j,\, i \in N, \quad \alpha + \beta = 1, \quad \alpha,\, \beta \in [0,\, 1]. \end{aligned} \tag{6}$$

Algorithm 1: Fuzzy Representation Inference

Data: An uploaded video from a user
Result: An optimized server to transcode the video

```
S_li: Estimation time of transcoding at server i;
Calculating loads of server i using (1);
Estimating the total load of server i using (2) and (3);
while not end of transcoding request do
    QueuingTranscodingJob(); push a new transcoding job into a queue;
    for job in the queue do
        EstimateTranscodingTime(); //for each video in the queue using (4) and (5)
    end
    SolvingOptimizationProblem(); // using (6);
    // get an ordering list of transcoding jobs.
    ExecutingTranscoding(); // the jobs in the ordering queue;
end
```

In the evaluation metrics, we consider two factors of the client side at the view of the response of a system for a given request from a user. First, start latency (S_l) describes the period between the time when a video source starts uploading to the cloud and the time when the video is available for playback. We define S_{lj} as the start latency on server jth. Second, the number of representations (N_r) represents a quality range which the cloud can provide for a user. Suppose we have $n \in \mathbb{N}$ servers in a cloud. A server $j \in \mathbb{N}$, $j \le n$ has load measurement with six parameter described above, is m_{ij}, $i \in \{1, 2, \ldots, 6\}$. Maximum load of the server is L_j, and maximum network measurement is N_j. Our goal is to minimize the total start latency with limited cloud computation resource and network connection throughput. Reasonably, the time of transcoding process increases if server load increases and the network throughput decreases. Therefore, we form optimization problem as shown in (6) with α, β are factors which describe relations between server loads and network states with transcoding time. As such, we can solve the optimization problem with a solution which refers mostly to whether the network or local side. For example, we can incline to the network side by assigning $\alpha = 0.4 > \beta = 0.6$. The solution of this problem is an ordering list of transcoding jobs which reduce total transcoding time. In conclusion, we summarize the proposed method in Algorithm 1.

Table 1 List of video source file in the experiment

Source video	Max resolution	Duration
(Paddy_Sun)_Sunflower	360p	3m30s
SIXTEEN_Major_A_Happy	720p	2m31s
Avicii___Wake_Me_Up	720p	4m32s
Britt Nicole - Ready or Not	720p	2m58s
Craig David - Insomnia	720p	3m40s
4K_Hawaii Drone Footage	2160p	11m10s
NewYork In_4K	2160p	4m41s

5 Experiment and discussion

In this section, we implemented a cloud system using Docker with the following descriptions. First, we installed a DASH streaming system and made a Docker container which con-

Table 2 Transcoding bitrate information

Video standard	Resolution	Bitrate (bits)
144p	80 × 144	196,217
240p	134 × 240	295,360
360p	202 × 360	394,284
480p	270 × 480	493,986
720p	404 × 720	1,478,541
1080p	606 × 1080	2,934,266
1440p	810 × 1440	5,842,639
2160p	3840 × 2160	21,400,447

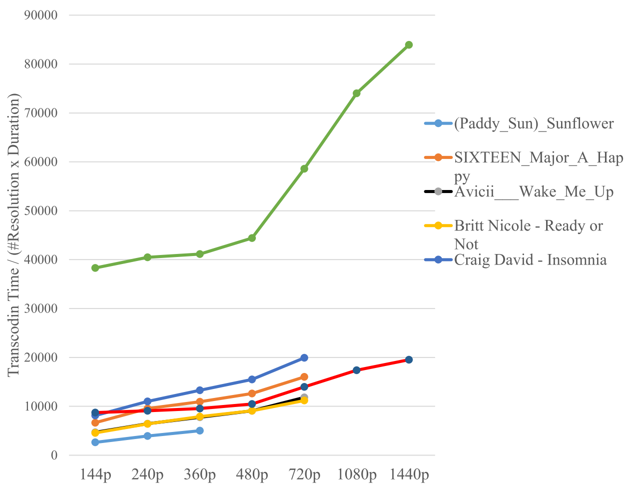

Fig. 9 Transcoding time for each representation of each video

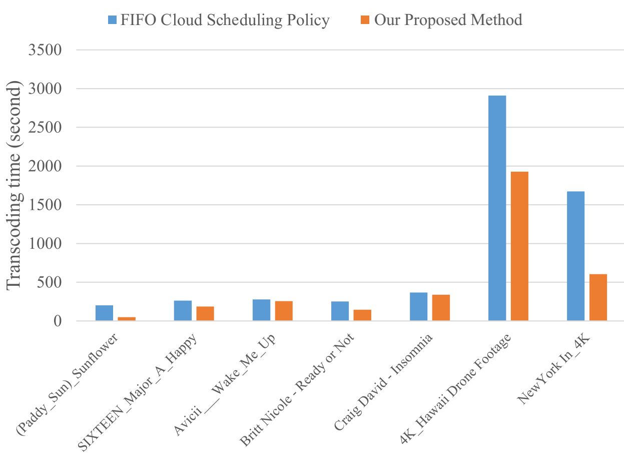

Fig. 10 Multimedia transcoding time comparison

tains the DASH server as well as the runtime environment. The DASH implementation is a Node.js-based [21,22] system. Secondly, several software applications are installed to support adaptive streaming. For example, we used the Fast Forward Motion Picture Experts Group (FFMPEG) software [23] to transcode a multimedia video file into different presentations. Additionally, MP4Box software [24] generates an MPD file which contains information of a DASH video streaming. Moreover, we used dash.js open source [25] to build a client player interface which can change video representations along with the fluctuation of throughput. In the system, we implemented our proposed optimization method in a management server, where we process all requests from clients to solve the optimization problem.

We tested with six video source files obtained from YouTube as shown in Table 1. The detail of transcoding bitrate is depicted in Table 2 with maximum vertical resolution up to 4K. We implemented a program in servers which transcode a source video from 144p to the maximum vertical resolution of a given video. By doing this way, we ensure that users can watch streaming video in different qualities depending on the quality of the network.

Transcoding time requires for a video, is proportional to a number of resolution and video duration. As shown in Fig. 9, a video named "4K_Hawaii Drone Footage" increased transcoding time dramatically compared to other videos with lower duration length and fewer resolutions.

As addressed in Sect. 2.1, we can run Docker container directly without installing any addition libraries or runtime environment. In the illustration of our approach, we compared the proposed method with first in first out (FIFO) schedule policy [26] in a cloud environment. We ran three DASH streaming servers in which one server is a management server and other two act as data servers. As a result in Fig. 10, our method reduced transcoding time significantly compared to existing methods. Besides, Figs. 11 and 12 show

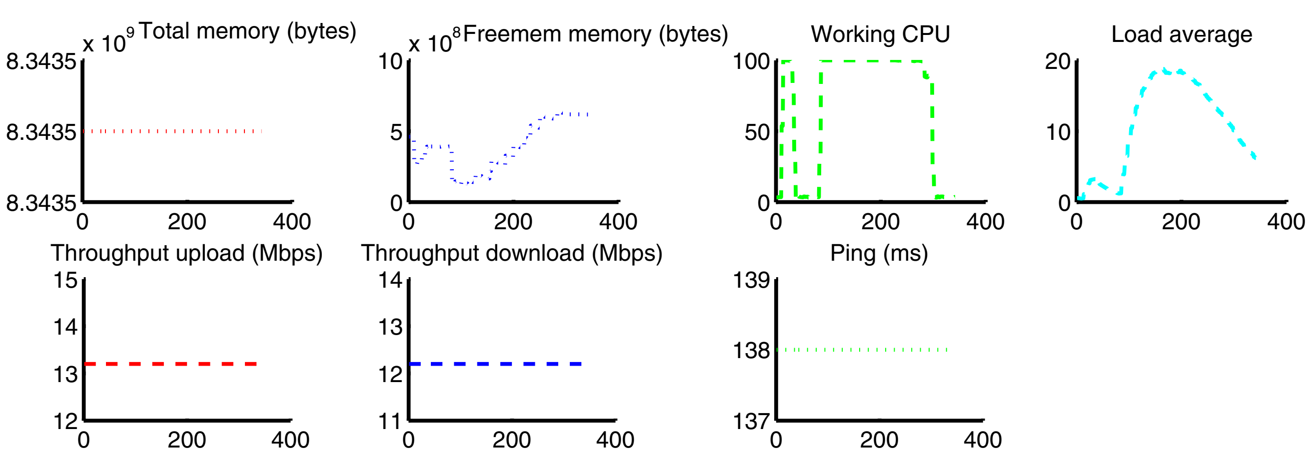

Fig. 11 Performance of a server using FIFO schedule policy

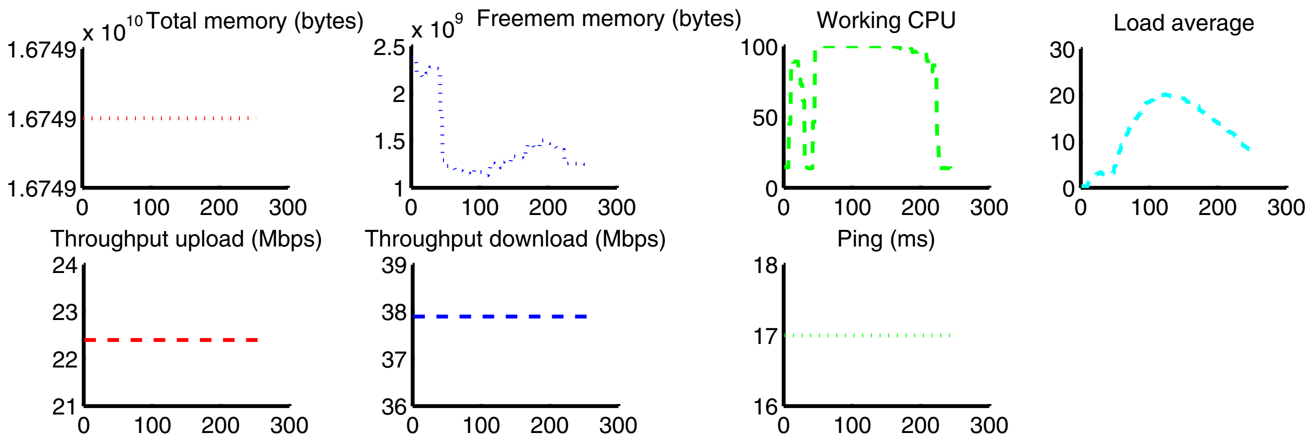

Fig. 12 Performance of a server with seven given parameters using our proposed method

a comparison between our method and the FIFO in the same test of transcoding three video files. Our approach completed the task within 300 s (6 min) meanwhile the FIFO method spent 400 s completing the task. Recall that, by using Docker, we are able to join and connect computers into one which also called Docker Swarm [8]. As shown in Fig. 12, total memory and throughput are merged to have more resource providing for a task. While doing transcoding task at servers, the FFMPEG software used 100% of total CPU capacity which could speed up and complete the job as quickly as the system can.

6 Conclusion

In this article, we have focused our research on the performance enhancement of DASH streaming system in a cloud environment. By finding an ordering list of servers for transcoding multimedia files, we proposed a method which reduced the total transcoding time of video source files. First, we used moving average to weaken the impact of the continuous change of the measurement values. Then, we proposed the scheduling algorithm for transcoding video files. As a result, it showed that our method reduced the transcoding time up to 30% compared to existing research.

In the future research, we intend to investigate in network functionalization to manage a cloud streaming system and control follow of packages efficiently.

Acknowledgements This work was supported by 'The Cross-Ministry Giga KOREA Project' Grant from the Ministry of Science, ICT and Future Planning, Republic of Korea (GK16P0100, Development of Tele Experience Service SW Platform based on Giga Media).

References

1. World Wide Web Consortium, et al.: Internet Users. Internet Live Stats (2015)
2. Rittinghouse, J.W., Ransome, J.F.: Cloud Computing: Implementation, Management, and Security. CRC Press, Boca Raton (2016)
3. Puthal, D., Sahoo, B., Mishra, S., Swain, S.: Cloud computing features, issues, and challenges: a big picture. In: 2015 International Conference on Computational Intelligence and Networks (CINE), pp. 116–123. IEEE (2015)
4. Seufert, M., Egger, S., Slanina, M., Zinner, T., Hobfeld, T., Tran-Gia, P.: A survey on quality of experience of HTTP adaptive streaming. IEEE Commun. Surv. Tutor. **17**(1), 469–492 (2015)
5. Wang, X., Chen, M., Kwon, T.T., Yang, L., Leung, V.C.: AMES-Cloud: a framework of adaptive mobile video streaming and efficient social video sharing in the clouds. IEEE Trans. Multimed. **15**(4), 811–820 (2013)
6. Jin, Y., Wen, Y., Westphal, C.: Optimal transcoding and caching for adaptive streaming in media cloud: an analytical approach. IEEE Trans. Circuits Syst. Video Technol. **25**(12), 1914–1925 (2015)
7. Joy, A.M.: Performance comparison between Linux containers and virtual machines. In: 2015 International Conference on Advances in Computer Engineering and Applications (ICACEA), pp. 342–346. IEEE (2015)
8. Boettiger, C.: An introduction to Docker for reproducible research. ACM SIGOPS Oper. Syst. Rev. **49**(1), 71–79 (2015)
9. Pourazad, M.T., Doutre, C., Azimi, M., Nasiopoulos, P.: HEVC: the new gold standard for video compression: how does HEVC compare with H.264/AVC? IEEE Consum. Electron. Mag. **1**(3), 36–46 (2012)
10. Hannuksela, M.M., Rusanovskyy, D., Su, W., Chen, L., Li, R., Aflaki, P., Lan, D., Joachimiak, M., Li, H., Gabbouj, M.: Multiview-video-plus-depth coding based on the advanced video coding standard. IEEE Trans. Image Process. **22**(9), 3449–3458 (2013)
11. Wu, J., Yuen, C., Wang, M., Chen, J.: Content-aware concurrent multipath transfer for high-definition video streaming over heterogeneous wireless networks. IEEE Trans. Parallel Distrib. Syst. **27**(3), 710–723 (2016)
12. Tekalp, A.M.: Digital Video Processing. Prentice Hall Press, Upper Saddle River (2015)
13. Boyce, J.M., Ye, Y., Chen, J., Ramasubramonian, A.K.: Overview of SHVC: scalable extensions of the high efficiency video coding standard. IEEE Trans. Circuits Syst. Video Technol. **26**(1), 20–34 (2016)
14. Ma, H., Seo, B., Zimmermann, R.: Dynamic scheduling on video transcoding for MPEG DASH in the cloud environment. In: Proceedings of the 5th ACM Multimedia Systems Conference, pp. 283–294. ACM (2014)
15. Krishnappa, D.K., Zink, M., Sitaraman, R.K.: Optimizing the video transcoding workflow in content delivery networks. In: Proceedings of the 6th ACM Multimedia Systems Conference, pp. 37–48. ACM (2015)

16. Liu, D., Zhao, L.: The research and implementation of cloud computing platform based on Docker. In: 2014 11th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), pp. 475–478. IEEE (2014)
17. Ismail, B.I., Goortani, E.M., Ab Karim, M.B., Tat, W.M., Setapa, S., Luke, J.Y., Hoe, O.H.: Evaluation of Docker as edge computing platform. In: 2015 IEEE Conference on Open Systems (ICOS), pp. 130–135. IEEE (2015)
18. Abdelbaky, M., Diaz-Montes, J., Parashar, M., Unuvar, M., Steinder, M.: Docker containers across multiple clouds and data centers. In: 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), pp. 368–371. IEEE (2015)
19. Barik, R.K., Lenka, R.K., Rao, K.R., Ghose, D.: Performance analysis of virtual machines and containers in cloud computing. In: 2016 International Conference on Computing, Communication and Automation (ICCCA), pp. 1204–1210. IEEE (2016)
20. Aparicio-Pardo, R., Pires, K., Blanc, A., Simon, G.: Transcoding live adaptive video streams at a massive scale in the cloud. In: Proceedings of the 6th ACM Multimedia Systems Conference, pp. 49–60. ACM (2015)
21. Madsen, M., Tip, F., Lhoták, O.: Static analysis of event-driven Node.js JavaScript applications. In: ACM SIGPLAN Notices, vol. 50, pp. 505–519. ACM (2015)
22. Chaniotis, I.K., Kyriakou, K.I.D., Tselikas, N.D.: Is Node.js a viable option for building modern web applications? A performance evaluation study. Computing **97**(10), 1023–1044 (2015)
23. FFMPEG Team: http://FFmpeg.org (2013)
24. MP4Box G: Multimedia Open Source Project (2014)
25. Mueller, C., Lederer, S., Poecher, J., Timmerer, C.: Demo paper: Libdash-an open source software library for the MPEG-DASH standard. In: 2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), pp. 1–2. IEEE (2013)
26. Li, J., Ma, T., Tang, M., Shen, W., Jin, Y.: Improved FIFO scheduling algorithm based on fuzzy clustering in cloud computing. Information **8**(1), 25 (2017)

Linh Van Ma is currently an M.S. Candidate at the Smart Mobile and Media Computing Laboratory, School of Electronics and Computer Engineering, Chonnam National University, South Korea. He received his B.S. Degree from the School of Applied Mathematics and Informatics, Hanoi University of Science and Technology, Vietnam in 2013. He was an Engineer at the Samsung Vietnam Mobile R&D Center in 2013. His major interests are in the research areas of Mobile Cloud Computing and Applied Mathematics in Network Communication.

Jaehyung Park received his B.S. in computer science from Yonsei University, Korea, in 1991, and his M.S. and Ph.D. in computer science from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 1993 and 1997, respectively. From 1997 to 1998, he was with the Center for Artificial Intelligence Research (CAIR) in KAIST. From 1998 to 2002, he was with the Network Laboratory in ETRI. Since 2002, he has been with the faculty of Chonnam National University, Korea, where he is currently an associate professor with the School of Electronics and Computer Engineering. His research interests are Internet Routing, Multicast Routing, Network Security, Wireless Mesh Networks, and Wireless Mesh/Ad-hoc Networks.

Jiseung Nam received his Ph.D. degree in computer science and engineering from the University of Arizona, the USA in 1992. He worked as a researcher at the Chonnam National University Center for Information and Communications Specialization, Gwangju, S. Korea from 1999 to 2001. Currently, he is a professor in Chonnam National University, Gwangju, South Korea. His research interests include Communication protocol, Internet real-time services, and routing.

Jonghyun Jang received his Ph.D. Degree in Computer Science and Engineering from Hankuk University of Foreign Studies, Yongin, Republic of Korea, in 2004. Since 1994, he has been with ETRI, where he is currently a Principal Member of the Research Staff as well as a Team Leader of the Real and Emotional Sense Platform Research Section. He has worked on several projects for the development of programming environments and PCS since 1994. His research interests are real-time middleware for telecommunication systems, home networking systems, and real sense media services.

Jinsul Kim received the B.S. Degree in Computer Science from the University of Utah, Salt Lake City, Utah, USA in 2001, and the M.S. and Ph.D. Degrees in Digital Media Engineering, Department of Information and Communications of the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea in 2005 and 2008. He worked as a Researcher in the IPTV Infrastructure Technology Research Laboratory, Broadcasting/ Telecommunications Convergence Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea from 2005 to 2008. He worked as a Professor in the Korea Nazarene University, Chon-an, Korea from 2009 to 2011. Currently, he is a Professor in Chonnam National University, Gwangju, Korea. His research interests include QoS/QoE, Measurement/ Management, IPTV, Mobile IPTV, Smart TV, Multimedia Communication and Digital Media Arts.