Cost-efficient and QoS-aware Content Management in Media Cloud: Implementation and Evaluation

Guanyu Gao, Yonggang Wen, Weiwen Zhang, Han Hu
School of Computer Engineering
Nanyang Technological University
Email: {ggao001, ygwen, wzhang9, hhu}@ntu.edu.sg

***Abstract*—Adaptive bitrate streaming has been proposed to encode video contents into multiple versions for device heterogeneity and changing network conditions. This solution, however, could consume enormous computing and storage resource. In fact, only a small fraction of videos are frequently requested. Thus, caching multiple versions for unpopular contents is not cost efficient. In this paper, we design a cost-efficient and QoS-aware content management system for video streaming. The system consists of a set of streaming servers and a computing cluster, where streaming servers can cache video contents or transcode them in real time, and the computing cluster can perform transcoding tasks on behalf of streaming servers. Based on this architecture, to provide cost-efficient and QoS-aware video service, first, we design a cost-efficient content cache management module to minimize the operational cost, by dynamically determining whether a segment should be cached or transcoded on fly according to their popularity. Second, to reduce transcoding latency, we design a QoS-aware transcoding task delegation module to determine whether a transcoding task in streaming server should be delegated to the computing cluster according to the streaming server's workload. We implement the system and evaluate the performance in a real environment. The results demonstrate that our method can greatly reduce the operational cost and guarantee the QoS in providing video services.**

I. Introduction

With the rapid growth of video sharing service, multimedia big data [1], and the advancement of the wireless technology, online video has become an important medium for people to have entertainment, obtain information, and connect with each other. However, due to the heterogeneous devices and changing network conditions, it has been a challenge for service providers to deliver reliable video service and provide a high quality of experiences to users [2]. Various screen sizes and device capacities would result in different requested bitrates and resolutions. In addition, the changing network condition can also affect the quality of the steaming service. Adaptive bitrate (ABR) streaming [3] is a popular technique to tackle such problems. In ABR, a video content is encoded into multiple versions in varying bitrates and resolutions, and each of the versions is divided into multi-second segments. When a user is viewing videos, it can detect the device capacity and network bandwidth to adjust the quality of the requested video segment accordingly. As a result, it can always deliver the best possible quality video to users.

Nevertheless, the ABR solution could consume a large amount of computing and storage resource, due to the huge demand of video contents. First, since the transcoding operation is rather CPU-intensive, it would take a long time to encode a video content into multiple versions. This situation is further stressed when users continuously publish new video contents. For instance, the new published video contents in YouTube during one minute are 100 hours [4]. Thus, it needs to deploy a large number of computing servers to perform transcoding for ABR. Second, the storage space consumption will increase, since ABR caches multiple versions of a video content in the server. However, among the sheer volume of video contents, only a small fraction of them are frequently requested by users, following the Pareto principle [5]. Meanwhile, some categories of the video content (e.g., news) would quickly fade out after their publish and become seldom requested [6]. As a result, it can be a waste of resource if all versions of the videos are cached in the system during their whole life time, especially when they have become unpopular and seldom requested.

Both computation and storage consumption in ABR would result in operational cost for service providers. To provide the ABR service in a cost-efficient manner, [7] considers the tradeoff between computation and storage cost for storage efficiency. In [8], we propose a partial transcoding scheme for ABR to minimize operational cost. However, both of these works do not consider the impact of transcoding latency, which is critical for video streaming. Meanwhile, the previous works still lack of performance analysis of such systems under a real system implementation. In this research, we design and implement a cost-efficient and QoS-aware content management system for ABR-based video streaming. The system consists of a set of streaming servers and a computing cluster. First, each streaming server can cache a subset of the video segments. If the requested segment is in the cache, then the streaming server can respond to the user directly; otherwise, the streaming server will encode it from the original bitrate segment in real time. Second, when the streaming server's computing capacity becomes over-loaded, the transcoding task can be delegated to the computing cluster, which has sufficient computing resource, for transcoding to reduce the latency.

Based on this system architecture, we design the system with the principle of providing cost-efficient and QoS-aware video streaming service. To this end, first, we design a cost-efficient content cache management module, which can dynamically determine video segments of each version cached in the streaming servers. We aim to minimize the cost by determining whether a segment should be locally cached or transcoding on fly, while guaranteeing the average request latency and satisfying the storage capacity constraint.

978-1-4673-6432-4/15/$31.00 ©2015 IEEE

Second, we design a QoS-aware transcoding task execution module, which can determine whether the transcoding task should be delegated to the computing cluster according to a threshold policy. The threshold is based on the maximum latency acceptable to a user request.

We implement the system and conduct extensive measurements in terms of resource utilization and system performance. The results demonstrate that 1) our proposed method can save 40% of the storage space and improve the streaming server's idle computing resource utilization; 2) our proposed method can reduce the operational cost for 30% in providing video service, compared to the method of caching all segments; 3) our proposed method can reduce the transcoding latency under the transcoding task delegation module.

The rest of this paper is organized as follows: Section II presents measurements and the motivation for our work. Section III introduces the design of our proposed system. Section IV gives the principles for cost-efficient video content management. Section V illustrates the system implementation and evaluation. Section VI concludes this paper.

II. Measurements and Observations

In this section, we illustrate the user viewing behavior in previous measurements, and measure the impact of ABR on the server resource consumption in a real environment, which are the motivations for our system design.

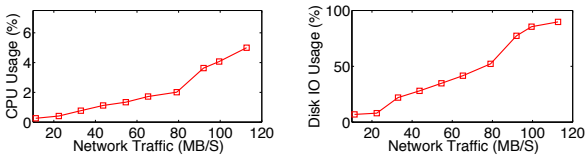
A. User Partial Interest on Video Content

Previous measurements have demonstrated that most of the video contents are not very popular; nearly 80% of the total views only come from 10% of the most popular videos [5]; and for 60% of the videos, only less than 20% of their duration is viewed [9]. The results show that among the sheer volume of the video contents, only a small part are frequently requested by users. It also indicates that in ABR, caching multiple versions for each video content without considering their popularity cannot be cost-efficient, especially for those seldom requested contents.

B. Streaming Server Utilization

Since the storage space of the streaming server is limited, the number of unique video contents that can be stored will decrease if some contents are cached with many versions. As measured in our deployment, the number of unique video contents in a server will decrease to nearly 40% if each content is cached with 5 versions [1]. As such, the operational cost of the cached content will increase severalfold. On the contrary, the utilization of a streaming server will decrease, since less unique contents are cached in the server. Therefore, the server cannot be fully utilized to serve as many requests as possible. These observations indicate that although the storage space of the streaming server could be large, we still need to reduce the storage consumption of each content for ABR streaming. Otherwise, the number of contents stored on the server will be reduced and the utilization of the server is low.

[1]The resolutions of the 5 versions: 1080p, 720p, 480p, 360p and 240p.



(a) CPU Usage (b) Disk IO Usage

Fig. 1. CPU usage and disk IO usage under varying network traffic. The video contents in the streaming server are randomly requested.

C. Streaming Server Resource Consumption Pattern

The streaming server in the media cloud is responsible for rendering the user requested video segment, therefore, most of the operations in the streaming server are IO operations, e.g., reading video files from disk and sending data to users [10]. In our deployment we measure the relations among the CPU usage, disk IO usage and network traffic under the condition that video contents in the streaming server are randomly requested [2]. The result is shown in Fig. 1. We can observe that the disk IO usage will increase with the network traffic, yet the CPU usage only increases slightly when the network traffic approaches to 110MB/S. These observations indicate that there is substantial idle computing resource in the streaming server, and using the idle resource for transcoding can improve its computing resource utilization.

The above measurements and observations motivate our research from three perspectives: 1) instead of caching multiple versions for each content, the content management should consider video popularity when determining whether the segments of a content should be cached; 2) to fully utilize a server's capacity to serve more requests, the content management should restrict storage space occupied by existing contents for the entry of other video contents; 3) the streaming server can leverage the idle computing resource to perform video transcoding while preventing excessive storage resource consumption.

III. System Design of Content Management in Media Cloud

In this section, we present the system design for our proposed video content management system in view of the observations we have found.

A. Generic System Architecture

Our proposed cloud-based streaming system is illustrated in Fig. 2. The system consists of three parts, including media outlets, streaming servers and computing cluster.

- The media outlets request a video segment in the best possible quality according to the current network condition and device capacity.
- The streaming servers response media outlets with the video segments in the requested bitrate. Due to its limited storage space, each streaming server stores a

[2]The server configuration: CPU, Intel Xeon E5-2620; Memory, 64GB; Network Interface, Gigabit Ethernet.

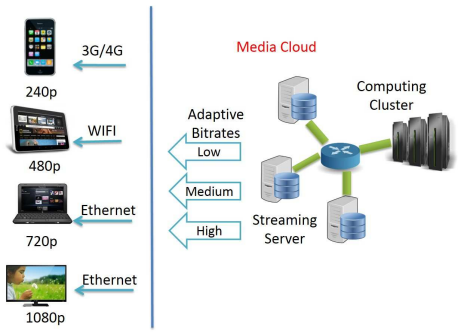


Fig. 2. The system architecture of the cost-efficient video content management system. The system consists of three parts, including media outlets, streaming servers, and computing cluster.

subset of all the video contents published by users and content providers.

- The computing cluster has sufficient computing capacity. Thus, it can execute some of the computing-intensive tasks in the media cloud (e.g., video transcoding, social media analytic, etc.).

B. Segment-based Content Management

To manage video contents in the media cloud cost-efficiently, we have a segment-based content management, which can dynamically change the versions of the contents cached in the system according to their popularity, based on the observation that users consume only a portion of them.

Each video content is encoded into several versions in varying bitrates and resolutions, and divided into multi-second segments. Under this model, only the version in the original bitrate is wholly kept in the system. For the versions in the other bitrates, only a fraction of the segments are kept in the system, which are determined their popularity. The video segments which are not present in the system will be transcoded on fly from the original bitrate segment when the user request arrives.

We consider two kinds of cost, including the storage cost and computing cost. The storage cost is incurred by keeping the cached video segments in storage space. The computing cost is incurred by encoding the original video segment into the user requested segment in case of cache miss. We aim to minimize the overall operational cost by determining which segments should be locally cached and which segments should be transcoded on fly. Specifically, for video segments that are frequently requested by users, it would be better to keep them in the storage, but would incur more storage cost. For video segments that are seldom requested, transcoding on fly can save storage cost, but would incur more computing cost.

C. Three-level System Design

We have a three-level design for the cloud-based content management as follows.

Level 1: If the requested segment is cached in its local storage, the segment will be read out from local storage and sent back to the user for consumption directly.

Level 2: If the requested segment is not cached in its local storage, cache miss will occur and a transcoding operation will be conducted to encode the video segment in the original bitrate into the user requested bitrate. Particularly, if the current workload of the streaming server is not high, the transcoding task can be conducted in the streaming server. As observed in our deployment, the streaming server has a low CPU usage even when the streaming traffic is high. Therefore, we can leverage the idle computing resource in the streaming server for transcoding.

Level 3: The workload of the streaming server may become high with many transcoding tasks to perform concurrently. In this case, if the incoming transcoding tasks are still conducted in the streaming server, it may incur long latency to the user, which would deteriorate the user's viewing experience. To guarantee the QoS, the transcoding task would be offloaded to the remote computing cluster for execution. The streaming server will first send the video segment in the original bitrate to the computing cluster. After the completion of transcoding, the transcoded segment in the user requested bitrate will be sent back to streaming server and then rendered to the user.

By the design of content cache management and transcoding scheme, we can achieve good experience for users while having low operational cost in the media cloud. In the next section, we will present the principles of cost-efficient video content management.

IV. Principles for Cost-efficient and QoS-aware Video Content Management

In this section, we develop two modules for the cost-efficient and QoS-aware video content management in the media cloud, including content cache management module and transcoding task delegation module.

A. Content Cache Management Module

The content cache management module determines which segments should be stored on streaming servers and which segments should be transcoded on fly by streaming servers when the user request arrives.

1) Operational Cost: We assume that the streaming server manages a set of M video contents, each of which will be encoded into N versions in varying bitrates. Each video file is split into 10-second long segments. There are two kinds of alternative costs that can be incurred.

First, for the video segment cached in the streaming server, it will take up storage space and incur storage cost. The storage cost is proportional to the segment size. We denote the file size of the ith video content, jth bitrate, and kth segment as $F(i,j,k)$, and the marginal price for storage space in a time slot as C^S. The storage cost for a cached segment during a time slot can be denoted as follows,

$$S(i,j,k) = F(i,j,k)C^S. \quad (1)$$

Second, for the video segment that is not locally cached, it will incur transcoding cost determined by the CPU time of the video transcoding operation and the request frequency of the segment. We denote the transcoding time for the ith video content, jth bitrate, and kth segment as $T(i,j,k)$, and

the request frequency of this segment in the time slot t as $\lambda^{(t)}(i,j,k)$. As such, the transcoding cost for a segment which is not locally cached in the time slot t can be denoted as

$$P^{(t)}(i,j,k) = \lambda^{(t)}(i,j,k)T(i,j,k)C^T, \quad (2)$$

where C^T is the marginal price for the CPU time.

Under our proposed content management model, a video segment is either cached in the storage or transcoded on fly on the streaming server. We define the binary variable $D^{(t)}(i,j,k)$ as the status of a video segment in the time slot t, where 1 represents that the segment is cached in the system, and 0 represents that the segment will be transcoded on fly. Therefore, the overall cost for managing all of the M video contents can be denoted as

$$O^{(t)} = \sum_{(i,j,k)\in\Gamma} (S(i,j,k)D^{(t)}(i,j,k) + P^{(t)}(i,j,k)\overline{D^{(t)}(i,j,k)}), \quad (3)$$

where Γ represents all of the segments for M video contents, and $\overline{D^{(t)}(i,j,k)}$ equals to $1 - D^{(t)}(i,j,k)$.

2) QoS Constraint: The video segment which is not cached in the system will be transcoded on fly when the user request arrives. However, it may incur latency to the user due to the execution time of the transcoding operation. With a less percentage of video segments cached in the system, the average request latency incurred by transcoding operation will become longer. Since the media cloud needs to provide the video service with guaranteed QoS, we need to control the latency incurred by transcoding. In this research, we adopt the average latency, which is the overall latency incurred by transcoding operations during one time slot divided by the total number of requests, to evaluate the QoS loss, represented as

$$L^{(t)} = \frac{\sum_{(i,j,k)\in\Gamma} \lambda^{(t)}(i,j,k)T(i,j,k)\overline{D^{(t)}(i,j,k)}}{\sum_{(i,j,k)\in\Gamma} \lambda^{(t)}(i,j,k)}. \quad (4)$$

3) Storage Capacity Constraint: The content management needs to consider the storage capacity constraint to prevent excessive storage resource consumption. Thus, when determining whether a segment should be cached in the system, it needs to ensure that the total file size of the cached segments will not exceed the allocated storage space. The total file size of the cached video segments can be denoted as

$$Q^{(t)} = \sum_{(i,j,k)\in\Gamma} F(i,j,k)D^{(t)}(i,j,k). \quad (5)$$

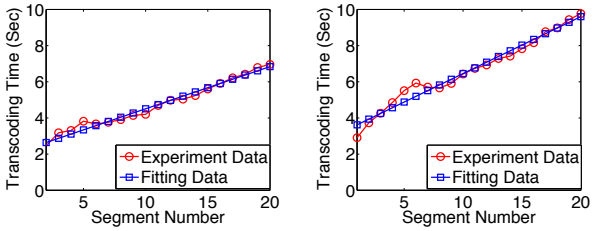
4) Problem Formulation: Our proposed content cache management module aims to reduce the overall operational cost for managing a set of video contents in the streaming server, while satisfying the storage capacity constraint and QoS requirement in each time slot. Mathematically, it can be formulated as the following optimization problem,

$$\min_{\vec{D}} \quad O^{(t)}, \quad (6)$$

$$\text{s.t.} \quad L^{(t)} \le L, \quad (7)$$

$$Q^{(t)} \le Q, \quad (8)$$

where L is the maximum endurable latency and Q is the allocated storage capacity for the set of video contents. The above problem is NP-hard, which cannot be solved optimally in polynomial time. We can use the Lagrange relaxation and subgradient method to obtain the approximate solution [11]. Due to the limited space, we omit the details of the solution.



(a) 426x240 (b) 1280x720

Fig. 3. Transcoding time for finishing varying numbers of video segments in the single targeted bitrate.

In our system implementation, the content cache management module works as follows: 1) collect user request information for each segment at the beginning of each time slot. 2) solve the optimization problem (6) to determine which segments should be cached in the streaming server and which segments should be transcoded on fly. 3) update the strategy obtained from Step 2 on the video contents.

B. Transcoding Task Delegation Module

The content cache management module can restrict the average request latency by controlling the percentage of video segments cached in the system. However, in some case when the user requests for uncached segments come in bursty, there may be no enough computing resource in the streaming server to conduct transcoding operations, which may incur great latency to users. In this case, the transcoding tasks will be delegated to the computing cluster for execution. As such, the streaming server needs to estimate its workload to determine whether the transcoding tasks should be delegated.

1) Workload Estimation: We evaluate the workload of the streaming server by estimating the transcoding time for finishing all of the segments waiting to be transcoded on the streaming server. We first measure the transcoding time for finishing a number of single targeted bitrate segments, the result of which is illustrated in Fig. 3. We observe that in the multi-core, multi-threading computing environment, the time for finishing all of the transcoding tasks is proportional to the number of video segments being transcoded, which can be denoted as

$$m_j(n) = a_j n + b_j \quad (n = 1, 2, 3, ...), \quad (9)$$

where $m_j(n)$ is the transcoding time for finishing n video segments in the jth targeted bitrate, and a_j and b_j are constants related to the CPU frequency and the number of cores.

However, users may desire to transcode the original video segments into different targeted bitrates. In this case, we cannot use the linear regression method to estimate the execution time for transcoding n original video segments into different targeted bitrates, since their execution time can be different. Instead, we use the linear combination method to estimate the

time for transcoding n video segments in different targeted bitrates,

$$\overline{E} = \sum_{j \in R} m_j(n) p_j, \tag{10}$$

where R is the bitrate set, and p_j is the percentage of the jth targeted bitrate video segments in the number of n being transcoded video segments.

2) Task Delegation: To avoid the long latency to the user, the transcoding task will be delegated to the computing cluster when there are too many transcoding tasks in the streaming server. We adopt a threshold policy to determine whether an incoming task should be delegated: 1) before system running, set a threshold for the maximum latency tolerant to each user; 2) when a transcoding task comes, query the current transcoding tasks in the streaming server and estimate the transcoding time by calculating Eq. (10); 3) If the estimated transcoding time exceeds the threshold, i.e., $\overline{E} > H$, then the incoming task will be delegated to the computing cluster.

V. System Implementation and Evaluation

In this section, we first introduce the implementation of our proposed cost-efficient video content management system in the real environment. Following that, we evaluate the performance of the system under various system settings.

A. System Implementation

1) Serving User Request: The implementation of our proposed system is illustrated in Fig. 4. We adopt the Apache and Web.py as webserver to serve user requests. When receiving a video segment request from the user, the webserver will first record the request information, including requested segment name and time, into Redis, which is a in-memory, key-value data store. And then, the webserver will search the requested segment in its local storage. If the segment is found, it will be rendered to the user directly. Otherwise, a transcoding operation will be performed.

2) Segment File Name Format: The file name of the video segment is in the following format,

VideoID_Width_Height_Bitrate_SegmentID,

which can ensure that its resolution, bitrate and corresponding original video segment can be detected from the file name directly. When the requested segment is not present in the storage, the streaming sever will extract the resolution and bitrate information from the requested segment name as transcoding parameters.

3) Transcoding Operation: Before the transcoding operation is conducted, the streaming server will first perform the workload estimation to determine whether this transcoding task should be delegated to the computing cluster. After that, the transcoding task will be either performed locally or delegated to the computing cluster.

4) Content Management: The streaming server periodically reads the segment request information from Redis and then executes the content cache management module to determine which segments should be cached according to their popularity.

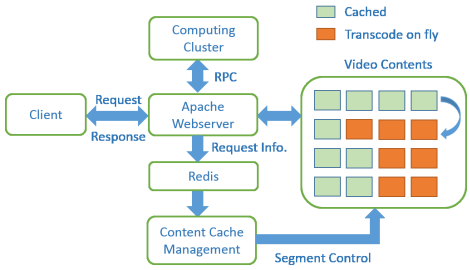


Fig. 4. The system implementation of the cost-efficient video content management system in the media cloud.

B. Experiment Settings

1) Server configuration: The streaming server configuration in our experiment is given as follows: CPU, Intel Xeon E5-2620; Memory, 64GB; Network Interface, Gigabit Ethernet. And we adopt the FFmpeg for video transcoding in the streaming server.

2) Storage cost and computing cost: We adopt the Amazon S3 Standard Storage and Amazon EC2 On-demand Instance to evaluate the storage cost and computing cost respectively. The storage price is $0.0300/GB/Month, and the computing price is $0.105/Hour.

3) Video versions: Each of the original video contents is encoded into 4 versions in different resolutions. The resolution of original video content is 1080p. The other versions are 720p, 480p, 360p and 240p. And all of the versions are split into 10-second video segments by using FFmpeg. The probability of each version being requested is equal.

4) Video request rate: We conduct the experiment with the real trace data in [12], which are the video view information collected from YouTube for 21 weeks. Particularly, the video request rate information is collected once a week. We adopt the user request rate for a video content in the previous period ($\lambda^{(t-1)}$) as the prediction of the request rate in the next period ($\lambda^{(t)}$), which cannot be known in advance.

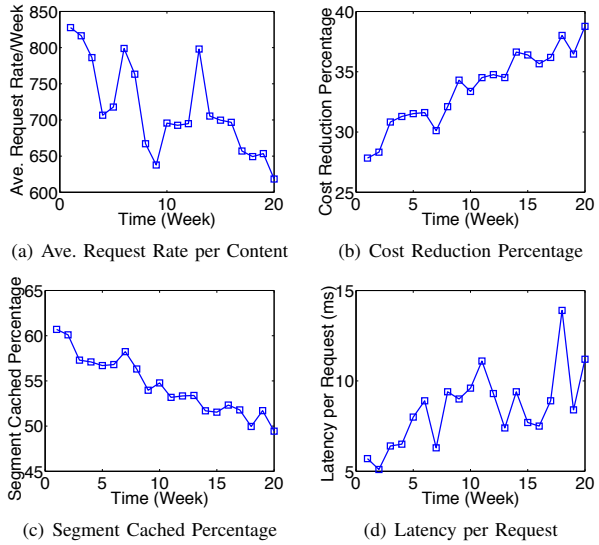
5) Segment requested probability: When starting a new video session, most of the users just view the first several video segments before aborting. By approximately fitting the data in [9], we define $F_i(k)$ to denote the probability in which the user would watch the ith video up to the kth segment,

$$F_i(k) = 1 - e^{-4.6\frac{k}{L_i}}, 1 \leq k \leq L_i,$$

where L_i is the number of segments for the ith content.

C. Performance under Real Request Rate Data

We measure the performance of our proposed content management module using the real YouTube trace data. Fig. 5(a) illustrates the trace data on the average request rate for the video contents during one week. First, we evaluate the cost reduction percentage of our proposed content management module, which is compared with the cost incurred by the method of caching all the segments in different bitrates. Fig. 5(b) shows that our proposed content management module can save more than 30% of the cost for most of the time, and the cost reduction percentage is growing over time, rising



(a) Ave. Request Rate per Content (b) Cost Reduction Percentage

(c) Segment Cached Percentage (d) Latency per Request

Fig. 5. The performance of the content management module under a real trace data over 20 weeks.

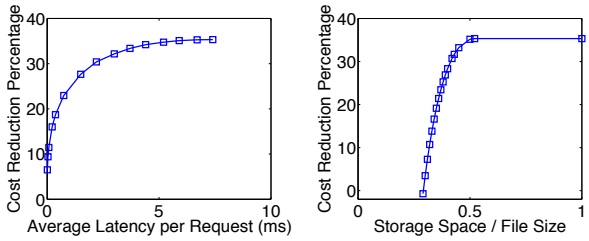
from nearly 30% to nearly 40%. This is because, most of the video contents are popular upon their initial publish, but the popularity will decrease over time such that it is not necessary to cache all the segments in different bitrates. Second, we observe that the segment cached percentage will decrease, as illustrated in Fig. 5(c). This is because, the average request rate for the video contents is basically decreasing over time (Fig. 5(a)). Third, as shown in Fig. 5(d), with more segments transcoded on fly, the latency incurred by transcoding operations will increase. This reflects the tradeoff between the operation cost and the latency in the content management.

This experiment indicates that our content management method, by dynamically adjusting the cached segments of each video content according to the video popularity, can have less operational cost than the method of caching all segments. Meanwhile, with less percentage of cached video segments, it can also save more storage space and increase the number of unique video contents cached in the streaming server.

D. Performance under constraints

The system may care more about the resource consumption and the service quality rather than the operational cost, when the resource is limited or a higher streaming quality is required. For instance, under the limited storage capacity, the system would restrict the storage space occupied by the existing video contents to introduce new video contents into the system. In addition, due to the service level agreement, it needs to control the average latency (cf. Fig. 5(d)) incurred by video transcoding to provide video service with a better user experience. Thus, we measure the impact of these two kinds of constraints on the cost reduction percentage, the results of which are given in Fig. 6.

From Fig. 6(a), we can observe that the cost reduction percentage is not very sensitive to the latency constraint. Even we restrict the average latency per request constraint from



(a) Average Latency Constraint (b) Storage Space Constraint

Fig. 6. The performance of the content management module under different constraints.

8ms to 2ms, the cost reduction percentage only drops slightly. However, if the average latency constraint is less than 1ms, the cost reduction percentage will drop dramatically. This is because, the fraction of the video segments cached in the system will increase sharply to avoid the latency incurred by video transcoding. In contrast, the cost reduction percentage is strongly related to the allocated storage space when the constraint is tight. We can observe from Fig. 6(b) that, when the storage space allocated to the video contents is close to 60% of the total file size of all video segments, the cost reduction percentage can achieve the optimal value. However, if the storage space is below this value, the cost reduction percentage will decrease dramatically. When storage space allocated to the video content is close to 30% of its total file size, it may even incur more operational cost than the method of caching all segments, due to frequently performed transcoding operations.

This experiment indicates that we need to consider the tradeoff among the operational cost, resource consumption and QoS for delivering video service.

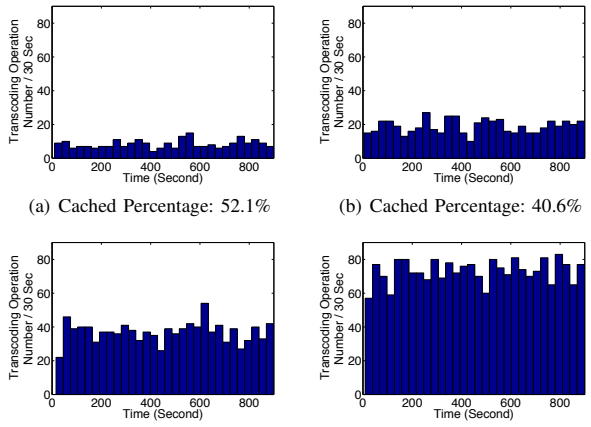
E. System Performance Metrics in Real Environment

We implement the system and measure its performance in the real environment. We select 10000 video contents stored in a streaming server. The total file size of these video contents, if all cached, is 1700GB and the incurred storage cost is $0.0708 per Hour. In case of transcoding task delegation, the average network traffic speed for transferring video segments between the streaming sever and the computing cluster is 110MB/S. We measure the performance metrics, including transcoding operation rate, CPU usage, operational cost and average response time per request.

TABLE I. PERFORMANCE METRICS UNDER VARYING ALLOCATED STORAGE SPACE

Allocated Storage Space (GB)	904	701	527	351
Segment Cached Percentage (%)	52.1	40.6	30.5	20.5
Response Time / Request (ms)	62.8	67.1	96.5	223
Operational Cost ($) / Hour	0.0492	0.0568	0.0764	0.1232

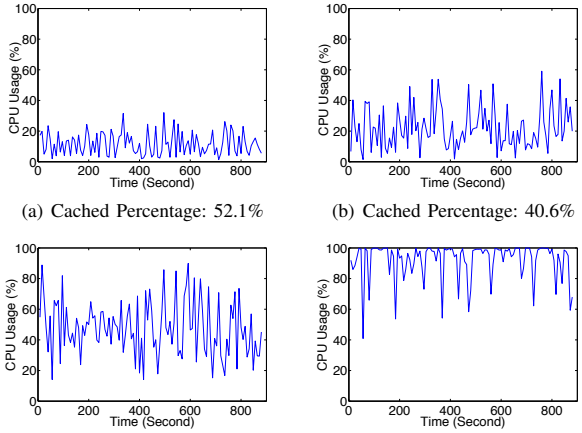
Table I illustrates the segment cached percentage, average response time and operational cost under varying allocated storage space. We can observe that our proposed content management method only needs to cache 52.1% of the video segments when the storage space is sufficient. Meanwhile, it can also save 30% of the operational cost, compared with the



(a) Cached Percentage: 52.1% (b) Cached Percentage: 40.6%

(c) Cached Percentage: 30.5% (d) Cached Percentage: 20.5%

Fig. 7. Transcoding Rate under Varying Segment Cached Percentage.



(a) Cached Percentage: 52.1% (b) Cached Percentage: 40.6%

(c) Cached Percentage: 30.5% (d) Cached Percentage: 20.5%

Fig. 8. CPU Usage under Varying Segment Cached Percentage.

method of caching all segments ($0.0708 per Hour). However, when we further restrict the storage space, the average response time for user requests will increase, since more transcoding operations will be conducted when there is no enough space to cache video segments. Meanwhile, the operational cost for managing these video contents will also increase.

Fig. 7 and Fig. 8 illustrate the transcoding operation rate and CPU usage under varying segment cached percentage, respectively. We can have two observations. First, we can strategically allocate the storage resource and computing resource to manage the tradeoff for minimizing the overall operational cost. Specifically, as the allocated storage space decreases, less video segments will be locally cached, resulting in less storage cost; however, the transcoding operation will be conducted more frequently (Fig. 7) and the CPU usage will increase (Fig. 8), resulting in more computing cost. Second, by leveraging the computing resource in the streaming server, it can greatly reduce the storage space occupied by video contents. Specifically, when the allocated storage space is 904GB (segment cached percentage is 52.1%), the CPU usage is still below 20% for most of the time. However, it can save about 50% of the storage space. As such, more video contents can be introduced into the streaming server, which can increase its resource utilization.

VI. Conclusion

In this research, we designed and implemented a cost-efficient and QoS-aware video content management system. The system consists of two modules. First, the content cache management module can save operational cost for streaming servers, by dynamically determining the segments of video contents cached in the streaming servers according to the video popularity. Second, the transcoding task delegation module can reduce the latency for video segment transcoding, by using a threshold policy to determine whether the transcoding task should be delegated to the computing cluster. With a real system implementation, we evaluated the system performance. Experiment results showed that the our designed system is cost-efficient, which can save 30% operational cost for providing video service compared to the method of caching all segments. Meanwhile, our system can improve the CPU utilization of streaming servers and reduce the latency by delegating the transcoding task to the computing cluster.

References

[1] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *Access, IEEE*, vol. 2, pp. 652–687, 2014.

[2] Y. Wen, X. Zhu, J. Rodrigues, and C. Chen, "Cloud mobile media: Reflections and outlook," *Multimedia, IEEE Transactions on*, vol. 16, no. 4, pp. 885–902, 2014.

[3] T. Stockhammer, "Dynamic adaptive streaming over http: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 133–144.

[4] "YouTube Statistics," https://www.youtube.com/yt/press/statistics.html, [Online; accessed Sep-2014].

[5] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 5, pp. 1357–1370, 2009.

[6] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," *Communications of the ACM*, vol. 53, no. 8, pp. 80–88, 2010.

[7] A. Kathpal, M. Kulkarni, and A. Bakre, "Analyzing compute vs. storage tradeoff for video-aware storage efficiency," in *Proceedings of the 4th USENIX Conference on Hot Topics in Storage and File Systems*, ser. HotStorage'12. USENIX Association, 2012, pp. 13–13.

[8] G. Gao, W. Zhang, Y. Wen, Z. Wang, W. Zhu, and Y. P. Tan, "Cost optimal video transcoding in media cloud: Insights from user viewing pattern," in *Multimedia and Expo (ICME), 2014 IEEE International Conference on*, July 2014, pp. 1–6.

[9] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "Youtube everywhere: Impact of device and infrastructure synergies on user experience," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 345–360.

[10] Z. Wang, L. Sun, C. Wu, W. Zhu, and S. Yang, "Joint Online Transcoding and Geo-distributed Delivery for Dynamic Adaptive Streaming," in *IEEE International Conference on Distributed Computing Systems (INFOCOM)*, 2014.

[11] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows: theory, algorithms, and applications," 1993.

[12] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of youtube videos," in *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*. IEEE, 2008, pp. 229–238.