QoE-Aware Video Storage Power Management Based on Hot and Cold Data Classification

Hwangje Han and Minseok Song
Department of Computer Engineering, Inha University
Incheon, Korea
iny27@naver.com,mssong@inha.ac.kr

ABSTRACT

Dynamically adaptive streaming over HTTP (DASH), the most common streaming technique, requires a video server to store all the transcoded versions, resulting in a lot of storage space, thereby consuming a significant disk power. A disk array can be divided into hot and cold zones to allow cold disks to be spun down, but this poses several questions such as (1) which video segments can be stored on the hot disks, (2) how to allocate video segments among the hot disks, and (3) how to handle requests to the cold disks. To address this, we propose three new algorithms; (1) a hot data classification algorithm to determine which segments should be stored on the hot disks, by taking segment popularity and quality-of-experience (QoE) into account, (2) a video segment allocation algorithm to balance workloads among the hot disks, and (3) a disk bandwidth allocation algorithm which determines the bit-rate of each segment with the aim of maximizing overall QoE. Experimental results show that our scheme can reduce the power consumption between 29% and 46% compared with the method of storing all the transcoded versions at the cost of 1.5% QoE degradation.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

Low-power systems, Video Servers, Hot and cold data

ACM Reference Format:
Hwangje Han and Minseok Song. 2018. QoE-Aware Video Storage Power Management Based on Hot and Cold Data Classification. In *NOSSDAV'18: 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video, June 12–15, 2018, Amsterdam, Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3210445.3210452

1 INTRODUCTION

Recent advances in network and system technologies have made it feasible to provide streaming services for a range of application areas, including user-created content (UCC) and personal game broadcasting. This increases the amount of Internet video traffic significantly, requiring a large server structure. For example, it has been reported that almost 5 billion videos are watched each day on YouTube [1].

Major streaming companies such as Netflix, Hulu and YouTube use dynamically adaptive streaming over HTTP (DASH) techniques [2, 3]. DASH divides each video into segments, each of which is transcoded into various bit-rate versions. This allows each client device to choose the most appropriate bit-rate version to match the network condition, and accurate prediction of network bandwidth is crucial for providing high level of quality-of-experience (QoE) [2, 3]; selecting a bit-rate lower than the actual bandwidth degrades video quality itself, whereas selection of a high bit-rate causes playback stall or buffering, which is one of the most annoying experiences to the clients.

DASH inherently requires a lot of transcoding operations. For example, Netflix offers video qualities between 1.5 Mbps and 25 Mbps, and it is known that 120 transcoding operations may be required for each video clip [4]. This also requires a lot of storage space to store transcoded versions, resulting in high disk power consumption to maintain many storage devices. A major consequence of this high power consumption is heat, which may accelerate an increase in the failure rate for disk drives [5]. What is worse, high energy consumption also impedes the expansion of servers, so it is essential to limit storage power consumption [5, 6].

Disks support multiple power modes [6]: in active mode the head is reading or writing data; in seek mode the head is seeking; in idle mode the disk spins at full speed but is processing no requests for data; and in standby, a disk stops spinning completely but consumes much less energy than in any other mode, but data on disk cannot be accessed. To minimize storage power consumption, a disk array can be divided into hot and cold zones to allow cold disks to be spun down [7]. However, data on the cold disks cannot be accessed in time, which makes hot and cold disk management important, but to the best of our knowledge, none of the previous studies have tackled this issue specifically for DASH-based video servers.

We propose a new disk power management scheme for DASH-based storage video servers. We first introduce the concept of QoE gain to express overall QoE for each bit-rate version by taking popularity and QoE into account. We then go on to propose three new algorithms: 1) a hot data determination algorithm to determine versions to be stored on the hot disks with the aim of maximizing QoE gain, 2) a hot disk allocation algorithm to balance workloads among the hot disks, and 3) a disk bandwidth allocation algorithm that allocates disk bandwidth to each request to limit power consumption consumed by a disk array with the aim of maximizing overall QoE.

NOSSDAV'18, June 12–15, 2018, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5772-2/18/06…$15.00
https://doi.org/10.1145/3210445.3210452

The rest of this paper is organized as follows: we review related work in Section 2 and propose storage and bandwidth allocation algorithms in Section 3 and Section 4, respectively. We assess our schemes in Section 5, and conclude the paper in Section 6.

2 RELATED WORK

Transcoding techniques for streaming are classified as online and offline schemes [4]; the online schemes transcode low bit-rate versions on request without storing them on disk, resulting in significant CPU computation, whereas the off-line schemes store all the transcoded versions, requiring a lot of storage space. To address this trade-off, Shen et al. [8] provided several strategies to combine online with offline transcoding schemes; Krishnappa et al. [4] presented a prediction model for future transcoding requests to prove the efficacy of online transcoding; Zhao et al. [9] presented an algorithm to determine which versions are stored on disk with the aim of minimizing CPU workloads by taking video internal popularity.

Video popularity follows a long-tail distribution, indicating that many versions may be requested only once or not at all [4]. To take advantage of this characteristics, Toni et al. [3] provided a transcoding parameter selection guideline by taking network characteristics into account, and Aparicio-Pardo et al. [10] improved it with the aim of maximizing average QoE specifically for live adaptive streaming. However, none of the works described above considered power issues.

To reduce CPU power consumption for transcoding operations, Zhang et al. [11] proposed a job dispatching algorithm to take different CPU power characteristics into account. Song et al. [12] proposed an algorithm that allocates a frequency and a workload to each CPU with the aim of minimizing power consumption while meeting all transcoding deadlines. However, no storage power issues were handled in these works.

To minimize storage power consumption, several cloud operators have developed cold storage platforms such as Facebook Cold Data Storage [13] and Microsoft Pelican [7]. Black et al. [7] presented several reports for the use of Pelican in the real field, especially focusing on archival disk usage. The main focus of these works, however, is to develop cold storage architecture without any algorithm provision that determines which files should be stored on the cold disks. To the best of our knowledge, this paper is the first systematic approach that aims at minimizing power consumption for DASH-based streaming systems based on hot and cold storage concept.

3 STORAGE ALLOCATION ALGORITHMS

3.1 Basic Idea

A commonly used method of energy saving is to transition a disk into standby mode after it has been idle for a while. However, this method of power saving is inapplicable to video servers because the time between consecutive disk requests generated by a typical server workload is too short and video data needs to be read continuously while it is being streamed [16].

To address this, we divide a disk array into hot and cold zones, and disks in the cold zone are spun down to save energy consumption, allowing only the data on the hot disk to be read. However, since the data on the cold disk is inaccessible, if a requested version is not available on the hot disks, then one of the lower bit-rate versions than the requested version on the hot disks should be delivered to clients. This leads to the following three issues, for which three algorithms will be provided:

(1) Hot data determination: Since only hot disks can be accessed, it is essential to determine which video segments should be stored on hot disks.
(2) Hot disk file allocation: A hot zone can be composed of multiple disks. It is therefore essential to distribute video segments across disks evenly; otherwise, although segments are stored on hot disks, they may not be accessed due to the shortage of disk bandwidth.
(3) Hot disk bandwidth allocation: When the requested segment is stored on the cold disks or the hot disks' bandwidth is insufficient, then one of the lower version than requested on the hot disks is streamed. This inevitably results in QoE degradation, so bandwidth allocation should minimize such overall QoE degradation.

3.2 Hot data determination

Let $p_{i,j}$ be the access probability of video version j of segment i, $V_{i,j}$, ($i = 1, ..., N^{\text{seg}}$ and $j = 1, ..., N^{\text{ver}}$), where $\sum_{i=1}^{N^{\text{seg}}} \sum_{j=1}^{N^{\text{ver}}} p_{i,j} = 1$, N^{seg} is the number of segments and N^{ver} is the number of versions. The popularity of segment i, p_i is then calculated as: $\sum_{j=1}^{N^{\text{ver}}} p_{i,j}$. Let L be the length of time in seconds for a video segment.

Let $S_{i,j}$ be the storage in MB for version $V_{i,j}$. Let S^{disk} be the size of a single disk. Let S^{total} be the total size of hot disks. Then, S^{total} can be calculated as follows:

$$S^{\text{total}} = N^{\text{hot}} S^{\text{disk}},$$

where N^{hot} is the number of hot disks. Let $Q_{i,j}$ be the video quality index for version $V_{i,j}$. The values of $Q_{i,j}$ can be easily derived by QoE measurement tools such as an SSIM tool [14]. Overall QoE gain, $G_{i,j}$ can be then defined to express the overall QoE achieved for the request to version $V_{i,j}$ as follows:

$$G_{i,j} = p_{i,j} Q_{i,j}.$$

Let $X_{i,j}$ be a binary variable to express whether version $V_{i,j}$ is stored on the hot disk or not; if $X_{i,j} = 1$, then version $V_{i,j}$ is stored on hot disks; otherwise it is stored on cold disks. The highest (original) version must be stored on the hot disks to transcode other versions. We assume that $\forall i$, $X_{i,1} = 1$, indicating that the lowest version for each video is stored on the hot disks. This guarantees that in any case a version with a bit-rate equal to or lower than the version requested by the client is sent to the client. We here define an optimization problem, called storage selection problem ($\mathcal{SSP}$) that finds $X_{i,j}$ for each transcoded version $V_{i,j}$, ($i = 1, ..., N^{\text{seg}}$, and $j = 2, ..., N^{\text{ver}} - 1$) to determine whether version $V_{i,j}$ can be stored on the hot disks.

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{N^{\text{seg}}} \sum_{j=2}^{N^{\text{ver}}-1} X_{i,j} G_{i,j} \\ \text{Subject to} \quad & \sum_{i=1}^{N^{\text{seg}}} \sum_{j=1}^{N^{\text{ver}}} X_{i,j} S_{i,j} \le S^{\text{total}}, \\ & X_{i,j} \in 0, 1, (i = 1, ..., N^{\text{seg}},\ j = 2, ..., N^{\text{ver}} - 1). \end{aligned}$$

1: An array of all the parameters $I_{i,j}$: A;
2: Temporary variable: $t \leftarrow 0$;
3: All the values of $X_{i,j}$ are initialized to 0;
4: **while** $A \neq \phi$ **do**
5: Choose a version $V_{i,j}$ with the highest value of $I_{i,j}$;
6: **if** $t + S_{i,j} \leq S^{\text{total}}$ **then**
7: $t \leftarrow t + S_{i,j}$;
8: $X_{i,j} \leftarrow 1$;
9: **end if**
10: $A \leftarrow A - \{I_{i,j}\}$;
11: **end while**

Figure 1: Hot data determination algorithm (HDA).

1: Temporary variable: $t_k \leftarrow 0$, $(k = 1, ..., N^{\text{hot}})$;
2: Array of segments in non-increasing order of p_i^{hot} values: A;
3: All the values of $Y_{i,k}$ are initialized to 0;
4: **while** $A \neq \phi$ **do**
5: Choose a segment i with the highest value of p_i^{hot};
6: Assign segment i to disk k with the smallest value of t_k;
7: $t_k \leftarrow t_k + S_i^{\text{hot}}$;
8: $Y_{i,k} \leftarrow 1$;
9: Remove segment i from A;
10: **end while**

Figure 2: Hot disk balancing algorithm (HBA).

$\mathcal{SSP}$ is a variant of 0/1 knapsack problem, where each object has a weight and a profit, and the problem involves selecting objects such that the total profit is maximized while satisfying the capacity constraints [15]. Since the 0/1 knapsack problem is NP-hard, it may not be possible to produce the optimal result within a reasonable time especially in our case where the number of segments is very large. We therefore develop a heuristic called hot data determination algorithm (HDA) as shown in Figure 1. In general, greedy algorithms exhibit good performance with knapsack problems [15]; therefore, we use a greedy approach, where a parameter, $I_{i,j}$ is defined for each version, $V_{i,j}$ as follows: $I_{i,j} = \frac{G_{i,j}}{S_{i,j}}$. Then HDA selects versions with the highest values of $I_{i,j}$ for the files to be stored on the hot disks subject to the total storage space, S^{total}.

3.3 Hot disk allocation

Since p_i represents the access probability of segment i, it is important to balance the sum of p_i values across the disks. To represent this, we introduce two variables, p_i^{hot} and S_i^{hot} to express the sum of overall probability and the storage size for video segment i stored on the hot disks, respectively as follows:

$$p_i^{\text{hot}} = \sum_{i=1}^{N^{\text{ver}}} X_{i,j} p_{i,j}, \text{ and } S_i^{\text{hot}} = \sum_{i=1}^{N^{\text{ver}}} X_{i,j} S_{i,j}.$$

Let $Y_{i,k}$ be a binary variable indicating whether video segment i is stored on disk k, $(Y_i = 1, ...N^{\text{hot}})$. For example, if $Y_{i,3} = 1$, then segment i is stored on disk 3. Then the sum of the access probabilities to disk k, S_k^{prob}, $(k = 1, ..., N^{\text{hot}})$ can be expressed as follows:

$$S_k^{\text{prob}} = \sum_{i=1}^{N^{\text{seg}}} Y_{i,k} p_i^{\text{hot}}.$$

To balance workloads among hot disks, we define an optimization problem, called hot data allocation problem ($\mathcal{HAP}$) that finds $Y_{i,k}$ for segment i as follows:

$$\text{Minimize} \quad |\max_k S_k^{\text{prob}} - \min_k S_k^{\text{prob}}|$$

$$\text{Subject to} \quad \forall k, (k = 1, ..., N^{\text{hot}}) \sum_{i=1}^{N^{\text{seg}}} Y_{i,k} S_i^{\text{hot}} \leq S^{\text{disk}},$$
$$Y_{i,k} \in 0, 1, (i = 1, ..., N^{\text{seg}},\ k = 1, ..., N^{\text{hot}}).$$

$\mathcal{HAP}$ is an allocation problem that aims at minimizing the maximum difference of popularities assigned to each disk to balance workloads among disks. We developed a hot disk balancing algorithm (HBA) based on worst-case allocation as shown in Figure 2. HBA first sorts an array of video segments in non-increasing order of p_i^{hot} values. Each sorted segment is then allocated one by one to the disk with the least workload.

4 BANDWIDTH ALLOCATION ALGORITHM

4.1 Algorithm description

When each segment is streamed, DASH allows each client to request a bit-rate version that best matches its network situation. If the requested version is located on the cold disks or disk bandwidth is insufficient, one of the lower versions than the requested is delivered to the client. Let N_k^{req} be the number of requests to the segments stored on disk k. Then the number of total request, N^{req} can be calculated as: $N^{\text{req}} = \sum_{k=1}^{N^{\text{hot}}} N_k^{\text{req}}$. Let $V(m)$ and $D(m)$ be the indices for the segment and the disk of request m, respectively. Let V_m^{req} be the version index for request m.

We introduce variables Z_m, $(m = 1, ..., N^{\text{req}})$, to express that the Z_mth version of request m is delivered to the client. We use a typical seek time model where a constant seek time T_s is required for one read of contiguous data [16]. Let tr be the transfer rata of the disk. Suppose that version $V_{i,j}$ has a bit-rate of $b_{i,j}$. Let $T_{m,j}$ be the service time to serve the jth version of request m, and can be calculated as follows:

$$T_{m,j} = T_s + \frac{b_{V(m),j} L}{tr}.$$

The total service time taken to read all the segments must not exceed the segment length, L to guarantee continuous delivery of video streams [16]. Therefore, the bandwidth allocation problem, $\mathcal{BAP}$ that aims at maximizing overall QoE subject to disk bandwidth constraint for each disk can be described as follows:

$$\text{Maximize} \quad \sum_{m=1}^{N^{\text{req}}} Q_{V(m),Z_m}$$

$$\text{Subject to} \quad \forall \text{ disk } k,\ \sum_{\forall m \text{ where } D(m)=k} T_{m,Z_m} \leq L.$$

$\mathcal{BAP}$ is a variant of multiple-choice knapsack problem, which is known to be NP-hard [15]. Moreover, since bandwidth allocation needs to be performed in real-time for each disk, time complexity must be considered. We thus developed a heuristic called bandwidth allocation algorithm (BAA) as shown in Figure 3 as follows:

Array of all the values of $R_{m,j}$'s: A;
Temporary variables: t_k, $(k = 1, ..., N^{\text{hot}})$;
$t_k \leftarrow \sum_{\forall m \text{ where } D(m)=k} T_{m,V_m^{\text{high}}}$, $(k = 1, ..., N^{\text{hot}})$;
$\forall\ m, Z_m \leftarrow V_m^{\text{req}}$;
while *TRUE* **do**
  if $\forall\ k, t_k \le L$ **then**
    break;
  end if
  Choose the lowest value of $R_{m,low}$ in the array A and removes $R_{m,low}$ from A;
  if $low < Z_m$ and $t_{D(m)} + T_{m,low} - T_{m,Z_m} > L$ **then**
    $t_{D(m)} \leftarrow t_{D(m)} + T_{m,low} - T_{m,Z_m}$;
    $Z_m \leftarrow low$;
  end if
end while

Figure 3: Bandwidth allocation algorithm (BAA).

(1) Ratio parameters for the jth version of request m, $R_{m,j}$, where $X_{V(m),j} = 1$, are defined as follows:

$$R_{m,j} = \frac{Q_{m,V_m^{\text{req}}} - Q_{i,j}}{T_{m,V_m^{\text{req}}} - T_{m,j}}.$$

(2) Let V_m^{high} be the the highest bit-rate version stored on the hot disks, which is less than or equal to the version index requested, V_m^{req}. An index variable for each request, Z_m is then initialized to V_m^{high}.

(3) The value of Z_m may be then decreased to decrease the bit-rate of segment when the total service time exceeds the segment length, L. The numerator of $R_{m,j}$ is the decrease in video quality, while the denominator is the increase in disk service time compared with the highest possible bit-rate. This means that the lowest value of $R_{m,low}$ in the set of $R_{m,j}$'s needs to be selected for the candidate version index, low, to minimize QoE degradation. Thus, the value of Z_m is decreased to low. This operation is repeated until $\forall$ disk k, $\sum_{\forall m \text{ where } D(m)=k} Z_m T_{m,j} \le L$.

4.2 Power analysis

Let P^{seek} be the power required during the seek phase, while P^{active} is the power required during the active phase, and P^{idle} is the power consumed during the idle phase. We then calculate the energy consumption of the hot disks for L seconds in each phase as follows:

(1) The total seek time is $\sum_{m=1}^{N^{\text{req}}} T_s$, so the energy required in the seek phase is

$$E^{\text{seek}} = \sum_{m=1}^{N^{\text{req}}} T_s P^{\text{seek}}.$$

(2) The time taken to read data is $\sum_{m=1}^{N^{\text{req}}} \frac{b_{i,Z_m} L}{tr}$, so energy required for reading data is calculated as

$$E^{\text{active}} = \sum_{m=1}^{N^{\text{req}}} \frac{b_{i,Z_m} L}{tr} P^{\text{active}}.$$

Table 1: Mapping between SSIM and QoE values [17].

$SSIM_{i,j}$	$Q_{i,j}$	Meaning
$SSIM_{i,j} \ge 0.99$	5	excellent
$0.95 \le SSIM_{i,j} < 0.99$	$25SSIM_{i,j} - 19.75$	good
$0.88 \le SSIM_{i,j} < 0.95$	$14.29SSIM_{i,j} - 9.57$	fair
$0.5 \le SSIM_{i,j} < 0.88$	$3.03SSIM_{i,j} + 0.48$	poor
$SSIM_{i,j} < 0.5$	1	bad

(3) If no disk activity is taking place, the disk is rotating without reading or seeking, which requires a power of P^{idle}. We calculate the total idle time during a length L by subtracting the seeking and reading times from L. Thus, the energy required in idle phase can be calculated as follows:

$$E^{\text{idle}} = (N^{\text{hot}} L - \sum_{m=1}^{N^{\text{req}}} (T_s + \frac{b_{i,Z_m} L}{tr}))P^{\text{idle}}.$$

By dividing the sum of energy consumption in each phase by the segment length L, the power consumed by the hot disk is calculated as: $\frac{E^{\text{seek}} + E^{\text{idle}} + E^{\text{active}}}{L}$.

5 EXPERIMENTAL RESULTS

5.1 Experimental Setup

We performed simulations to examine the effect of our scheme on the overall QoE and power consumption. We profiled video quality indices at each second for different resolutions of the five sample videos (Spiderman, The mummy, Transformer, War for the planet of the Apes and Wonder woman) using an SSIM tool [14] and then the SSIM value, $SSIM_{i,j}$ for each version $V_{i,j}$ are derived from these sample clips. However, QoE is usually represented by mean opinion scores (MOS) between 1 (bad) and 5 (excellent) [17], so the MOS values are used for $Q_{i,j}$, $(1 \le Q_{i,j} \le 5)$ as guided in Table 1 [17]. Each 1920×1080 video is assumed to be transcoded into six different-resolution versions: 1600×900, 1280×720, 1024×600, 854×480, 640×360 and 426×240. Then, the bit-rate of each YouTube HD-quality version was calculated using the video bit-rate calculator [18].

The length of each video segment is set to 2 seconds. The arrival of client requests is modeled as a Poisson process [19], with an arrival rate of 1/s. The access probability of each segment follows a Zipf distribution, where θ was set to 0.271 as profiled in a real VoD application [19]. A server is assumed to store 2000 video files which last between 2 and 3 hours. Users can terminate playback of a video before completion [20], so that the actual period of playback can be modeled by a Zipf distribution with $\theta = 0.2$ (see [20]). We used disk parameters of the Western Digital company in Table 2 [21]. Cold disks stay in standby mode, thus consuming standby power. We profiled power consumption for 24 hours, and the following two workloads are considered:

- HVP: High bit-rate versions are popular, ($\forall\ i$, $p_{i,1} = 0.1p_i$, $p_{i,2} = 0.1p_i$, $p_{i,3} = 0.2p_i$, $p_{i,4} = 0.3p_i$, $p_{i,5} = 0.3p_i$).
- LVP: Low bit-rate versions are popular, ($\forall\ i$, $p_{i,1} = 0.3p_i$, $p_{i,2} = 0.3p_i$, $p_{i,3} = 0.2p_i$, $p_{i,4} = 0.1p_i$, $p_{i,5} = 0.1p_i$).

Table 2: Disk specification for simulation [21].

Seek power	4.1W
Active power	4.1W
Standby power	0.4W
Idle power	2.7W
Transfer rate	147MB/s
Typical seek time	9ms

5.2 Algorithm efficiency

We validated each heuristic algorithm as follows:

(1) Hot data determination algorithm (HDA): We calculated the upper bound of the algorithm by relaxing integrality constraint imposed on $\mathcal{SSP}$ [15]. The maximum difference between HDA and this upper bound was found to be below 0.000004%, indicating that HDA yields a near-optimal solution to $\mathcal{SSP}$.
(2) Hot disk balancing algorithm (HBA): We examined the values of $\max_k S_k^{\text{prob}}$ and $\min_k S_k^{\text{prob}}$ for each algorithm execution. Table 3 tabulated the percentage difference for these two values, which clearly shows that HBA can balance overall popularity over each disk.
(3) Bandwidth allocation algorithm (BAA): To verify the efficacy of BAA, we developed a dynamic programming (DP) algorithm that runs for each disk as follows:
   (a) Let $V_{m,n}^{\text{qoe}}$ be the maximum QoE when n is the time length, in units of 0.1ms, required to process all the requests between 1 and m ($m = 1, \ldots, N_k^{\text{req}}$ and $n = 1, \ldots, L$). $V_{m,n}^{\text{qoe}}$ are all initialized to zero.
   (b) Next, $V_{1,n}^{\text{qoe}}$ is calculated as follows:

$$V_{1,n}^{\text{qoe}} = \max_{j \in \{j | T_{m,j} \le n\}} Q_{V(m),j}.$$

   (c) If $V_{m-1,n-T_{m,j}}^{\text{qoe}} > 0$, then $V_{m,n}^{\text{qoe}}$ can be updated using the following recurrence:

$$V_{m,n}^{\text{qoe}} = \max_{j=1,\ldots,V_m^{\text{req}}} \{\max(V_{m,n-1}^{\text{qoe}}, V_{m-1,n-T_{m,j}}^{\text{qoe}} + Q_{V(m),j})\}.$$

   (d) Based on the recurrence described above, we can derive a $V_{N_k^{\text{req}},L}^{\text{qoe}}$ value, which represents the optimal QoE value.

Although the DP algorithm can find the optimal QoE value, it takes a significant time to derive the solution, so it can be hardly applicable to our bandwidth allocation, which should be executed every L seconds to reflect workload changes. For example, on average, it takes 36.5s to execute DP, but 4.5ms to run BAA. To avoid unreasonably long running times, we executed DP every 30 minutes in the simulation, and Table 4 shows that the average percentage difference between BAA and DP. The difference is almost negligible, indicating that BAA produces a near-optimal solution within a reasonable time.

Table 3: Maximum percentage difference between $\max_k S_k^{\text{prob}}$ and $\min_k S_k^{\text{prob}}$ against S^{disk}.

(HVP, 2TB)	(LVP, 2TB)	(HVP, 3TB)	(LVP, 3TB)
0.01%	0.04%	0.01%	0.02%

Table 4: Average percentage difference between DP and BAA against S^{disk}.

(HVP, 2TB)	(LVP, 2TB)	(HVP, 3TB)	(LVP, 3TB)
0.004%	0.005%	0.3%	0.02%

5.3 Power and QoE results

Figures 4 and 5 show how the power consumption and QoE ratios vary with the number of hot disks compared with storing all transcoded versions, where the results indicate the following:

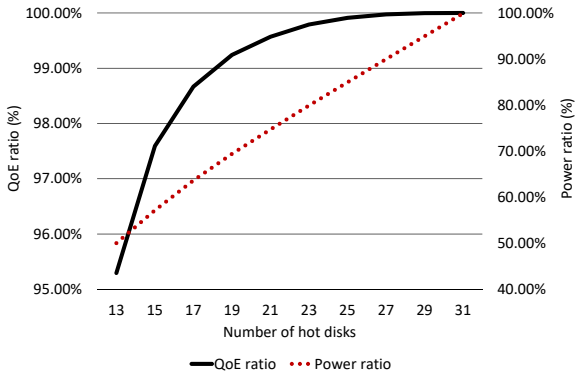
(1) The proposed scheme can effectively reduce the disk power consumption while minimizing QoE degradation. For example, when using a 2TB disk in an HVP workload, the disk power consumption is reduced by 50% at a 4.7% QoE degradation and by 31% at a 0.8% QoE degradation.
(2) Obviously, as the number of hot disks increases, more transcoded versions can be stored on the hot disk, thereby increasing the overall QoE, but this tails off.
(3) In LVP workloads, in contrast to HVP workloads, QoE degradation is negligible. For example, when the QoE is reduced by 1.2% in an LVP workload, the power consumption is reduced by 30% for a 3TB disk and by 43% for a 2TB disk.
(4) Compared to a 3TB disk, the power reduction effect is greater when using a 2TB disk. This is because more disks are used to store all the files when 2TB disks are used, which shows that the proposed scheme is more effective when using many disks.

6 CONCLUSIONS

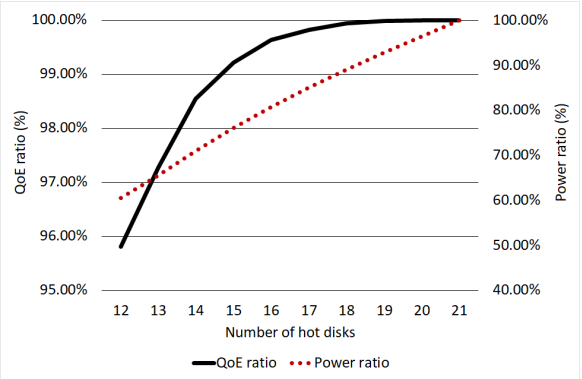
We have proposed a new scheme to minimize disk energy consumption in DASH-based video servers based on hot and cold storage concept. We introduced the concept of QoE gain to express overall QoE achieved by storing files on hot disks by taking version popularity into account. Based on this, we proposed and validated three new algorithms: 1) a hot data determination algorithm that determines the versions to be stored on hot disks, 2) a hot disk balancing algorithm that balances the workloads among hot disks and 3) a disk bandwidth allocation algorithm that determines the bit-rate of segments with the aim of minimizing QoE degradation subject to disk bandwidth constraint. Simulation results show that when the 1.5% QoE degradation is allowed, the proposed technique reduces the power consumption by 29% to 46% compared to storing all transcoded versions.

ACKNOWLEDGMENTS

This research was supported by Next-Generation Information Computing Development Program through National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT

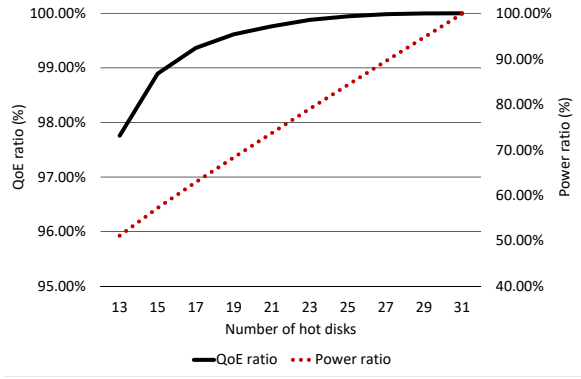


(a) $S^{\text{disk}} = 2\text{TB}$

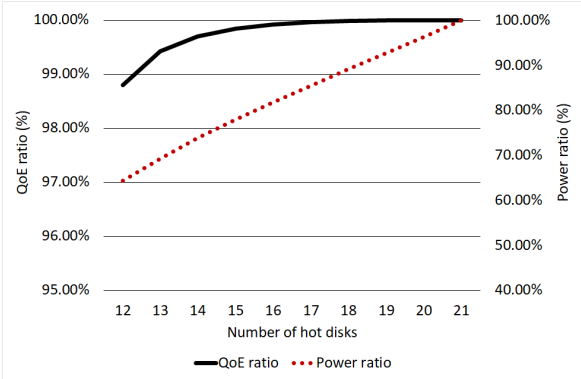


(b) $S^{\text{disk}} = 3\text{TB}$

Figure 4: QoE and power ratio in HVP workloads.



(a) $S^{\text{disk}} = 2\text{TB}$



(b) $S^{\text{disk}} = 3\text{TB}$

Figure 5: QoE and power ratio in LVP workloads.

(2017M3C4A7080248), and partly by the Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (NRF-2015R1D1A1A01058646). Minseok Song is a corresponding author.

REFERENCES

[1] Youtube bandwidth: terrabytes per day, http://blog.forret.com/2006/05/youtube-bandwidth-terabytes-per-day/.
[2] T. Stockhammer. Dynamic adaptive streaming over http: standards and design principles. In *Proceedings of the ACM Multimedia Systems Conference*, pages 133–144, 2011.
[3] L. Toni, R. Aparicio-Pardo, G. Simon, A. Blanc, and P. Frossard. Optimal set of video representations in adaptive streaming. In *Proceedings of the ACM Multimedia Systems Conference*, pages 271–282, 2014.
[4] D. Krishnappa, M. Zink, and R. Sitaraman. Optimizing the video transcoding workflow in content delivery networks. In *Proceedings of the ACM Multimedia Systems Conference*, pages 37–48, 2015.
[5] M. Dayarathna, Y. Wen and R. Fan. Data center energy consumption modeling: A survey. *IEEE communication surveys & tutorials*, 18(1):732–794, Jan. 2016.
[6] M. G. Khatib and Z. Bandic. PCAP: Performance -aware power capping for the disk drive in the cloud. In *Proceedings of the USENIX FAST*, pages 227–240, Feb. 2016.
[7] R. Black, A. Donnelly, D. Harper, A. Ogus, and A. Rowstron. Feeding the pelican: Using archival hard drives for cold storage racks. In *Proceedings of the USENIX Workshop on Hot Topics in Storage and File Systems*, pages 29–34, June 2016.
[8] B. Shen, S. Lee, and S. Basu. Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks. *IEEE Transactions on Multimedia*, 6(2):375–386, Apr. 2004.
[9] H. Zhao, Q. Zheng, W. Zhang, B. Du, and H. Li. A segment-based storage and transcoding trade-off strategy for multi-version vod systems in the cloud. *IEEE Transactions on Multimedia*, 19(1):149–159, Jan. 2017.
[10] R. Aparicio-Pardo, K. Pires, A. Blanc, and G. Simon. Transcoding live adaptive video streams at a massive scale in the cloud. In *Proceedings of the ACM Multimedia Systems Conference*, pages 49–60, 2015.
[11] W. Zhang, Y. Wen, J. Cai, and D. Wu. Towards transcoding as a service in multimedia cloud: Energy-efficient job dispatching algorithm. *IEEE Transactions on Vehicular Technology*, 63(5):2002–2012, June 2014.
[12] M. Song, Y. Lee, and J. Park. Scheduling a video transcoding server to save energy. *ACM Transactions on Multimedia Computing Communications and Applications*, 11(2s):45, Feb. 2015.
[13] https://www.enterprisetech.com/2013/10/25 /facebook-loads-innovative-cold-storage-datacenter/.
[14] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, Apr. 2004.
[15] D. Pisinger. *Algorithms for Knapsack Problems*. PhD thesis, University of Copenhagen, 1995.
[16] M. Song, Y. Lee, and E. Kim. Saving disk energy in video servers by combining caching and prefetching. *ACM Transactions on Multimedia Computing Communications and Applications*, 10(1s):15, Jan. 2014.
[17] T. Zinner, O. Hohlfeld, O. Abboud, and T. Hossfeld. Impact of frame rate and resolution on objective qoe metrics. In *Proceedings of the IEEE International Workshop on Quality of Multimedia Experience*, pages 29–34, June 2010.
[18] https://toolstud.io/.
[19] A. Dan, D. Sitaram, and P. Shahabuddin. Dynamic batching policies for an on-demand video server. *Multimedia Systems Journal*, 4(3):112–121, 1996.
[20] S. Lim, Y. Ko, G. Jung, J. Kim, and M. Jang. Inter-chunk popualrity-based edge-first caching in content-centric networking. *IEEE Communication Letters*, 18(8):1331–1334, 2014.
[21] https://www.wdc.com/content/dam/wdc/website /downloadable_assets/eng/spec_data_sheet/2879-800002.pdf.