Video Management and Resource Allocation for a Large-Scale VoD Cloud

ZHANGYU CHANG and S.-H. GARY CHAN, The Hong Kong University of Science and Technology

We consider providing large-scale Netflix-like video-on-demand (VoD) service on a cloud platform, where cloud proxy servers are placed close to user pools. Videos may have heterogeneous popularity at different geo-locations. A repository provides video backup for the network, and the proxy servers collaboratively store and stream videos. To deploy the VoD cloud, the content provider rents resources consisting of link capacities among servers, server storage, and server processing capacity to handle remote requests.

We study how to minimize the deployment cost by *jointly* optimizing video management (in terms of video placement and retrieval at servers) and resource allocation (in terms of link, storage, and processing capacities), subject to a certain user delay requirement on video access. We first formulate the joint optimization problem and show that it is NP-hard. To address it, we propose Resource allocation And Video management Optimization (RAVO), a novel and efficient algorithm based on linear programming with proven optimality gap. For a large video pool, we propose a video clustering algorithm to substantially reduce the run-time computational complexity without compromising performance. Using extensive simulation and trace-driven real data, we show that RAVO achieves close-to-optimal performance, outperforming other advanced schemes significantly (often by multiple times).

CCS Concepts: • **Networks → Cloud computing**

Additional Key Words and Phrases: Distributed video-on-demand, optimization, resource allocation, linear programming, spectral clustering

ACM Reference Format:
Zhangyu Chang and S.-H. Gary Chan. 2016. Video management and resource allocation for a large-scale VoD cloud. ACM Trans. Multimedia Comput. Commun. Appl. 12, 5s, Article 72 (September 2016), 21 pages.
DOI: http://dx.doi.org/10.1145/2983638

1. INTRODUCTION

Cloud computing provides an attractive option to convert private infrastructure investment to daily operating expense. It substantially reduces the cost on purchasing geographically distributed servers and the risk of over-provisioning. Such paradigm has been used to deploy cost-effective distributed Internet services [Wu et al. 2011; Li et al. 2011; Adhikari et al. 2012; Aggarwal et al. 2011]. One of such services is Video-on-Demand (VoD), which has dominated the Internet traffic nowadays [Index 2013].

In a VoD cloud, the content provider can *rent* servers and bandwidth from one or multiple cloud service providers. Through this, resources can be rescaled in a timely and adaptive manner to satisfy fluctuating user demands while meeting user requirements

This work was supported, in part, by the Hong Kong Research Grant Council (RGC) General Research Fund (610713) and the National Natural Science Foundation of China (61472455).
Authors' addresses: Z. Chang and S.-H. G. Chan, Department of Computer Science and Engineering, the Hong Kong University of Science and Technology, Hong Kong; emails: {zchang, gchan}@cse.ust.hk.

DOI: http://dx.doi.org/10.1145/2983638

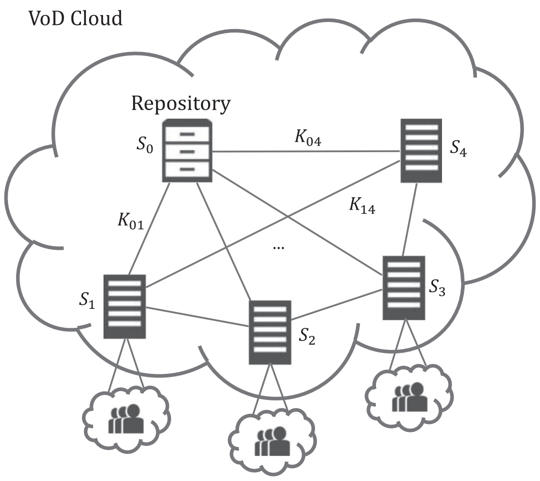

Fig. 1. A distributed and cooperative cloud architecture for VoD service.

(e.g., in video startup delay). We consider a Netflix-like VoD service with many titles. In such a network, user delay, defined as the time from his or her request to the instant of video playback, is an important quality of service (QoS) measure. As the scale of the service increases, how to minimize its deployment cost while meeting the user delay requirement becomes a critical issue.

We show in Figure 1 a typical cloud architecture for VoD. The cloud consists of a central server S_0 (repository) storing all the videos.[1] The content provider rents geographically distributed proxy (cloud) servers, labeled as S_1, S_2, S_3, and S_4, which have re-scalable resources (in terms of storage and processing/streaming capacities) and are close to user pools. We consider that videos are either entirely stored or not at all. While all the videos are stored in the central server, for cost-effectiveness the proxy servers may store only a fraction of them. Each user has a home (or local) proxy server to serve his or her video request. Due to geo-dispersed user groups, a video may have a different level of popularity at different home servers. If the content requested is stored locally (i.e., a hit), then the home server directly streams to its users from its storage. Otherwise (i.e., a miss), the home server requests the content from a remote server (which is either a proxy server or the repository), and the missed content is streamed via the home server to the users. To ensure service quality, dedicated end-to-end link capacity is rented to serve the misses between servers (labeled as K_{mn} in the figure). Furthermore, processing capacity (e.g., central processing unit (CPU), memory, streaming resources, etc.) of a server is needed to be rented in order to respond to all its remote requests. We will use "servers" to refer collectively both central and proxy servers. (Note that in this work though we mainly discuss in the context of video files of different streaming rates, our study can be similarly extended to multimedia files with different bandwidth requirements.)

The above collaborative cloud architecture achieves high resource scalability and cost-effectiveness. Because video popularity is skewed, we can achieve low storage cost by storing only a subset of the videos in each server. Moreover, the miss requests at a server may be served by other nearby proxy servers, hence saving link bandwidth from the repository. By allocating properly the amount of server storage, server processing

[1]In this article, we use "movie" and "video" interchangeably.

capacity, and link capacity, such an architecture can achieve tradeoff to minimize deployment cost while meeting a certain service requirement (on user delay). As an example, if storage cost is relatively low, then increasing server storage would lead to low overall deployment cost. On the other hand, if link cost is relatively low, server collaboration (and hence purchasing more link capacity) would lead to lower overall cost.

In contrast to volatile contents such as short video clips and user-generated content (UGC), the video popularity and traffic in a Netflix-like VoD system is relatively stable and predictable [Gong et al. 2010; Niu et al. 2012]. While videos cannot be replicated on-the-fly in the servers, they can be centrally *planned* on a longer time scale (say, days). This is similar to resource allocation, where capacities are rented to meet user delay requirements. That is to say, the content provider has to plan at intervals (at scheduled times or driven by system changes) the following two critical challenges:

—*Video management:* This involves content replication (CR) and server selection (SS). CR is to decide which video should be replicated in which proxy server(s). SS is to decide which remote server to serve the miss at a local server. Note that SS and CR are joint problems. Furthermore, as video popularity may differ at different servers, CR and SS decisions may not be the same for all the servers.

—*Resource allocation:* To meet the service quality, the content provider has to pay for the system resources in terms of link capacities allocated among servers, server processing capacity (to process and stream remote video requests), and server storage. In the VoD cloud, the deployment cost consists of two major components: (1) *server cost* due to the total storage and processing capacity at a server and (2) *link cost* due to the bandwidth capacity reserved and data transmitted between pairs of servers to serve the misses.

In the cloud paradigm, the content provider may dynamically adjust video management and the rented resources according to his predicted video demand over the time scale of interest (e.g., in days). As video management and resource allocation are interdependent decisions with the same time scale (i.e., video management can be optimized given resource allocation and vice versa), they need to be *jointly* optimized to minimize overall deployment cost, subject to a certain QoS requirement on user delay. This work is to address the joint optimization problem through the following:

—*Problem formulation and complexity analysis:* We study a novel problem for a VoD cloud, which is to jointly optimize video management and resource allocation to minimize the deployment cost given a certain user delay constraint. Our model is realistic, general, and comprehensive, capturing important system parameters on server cost, link cost, geographical heterogeneity of video access, and user delay as QoS constraint (previous work seldom considers all these parameters together). We show that such problem is NP-hard.

—*Resource allocation And Video management Optimization (RAVO): A novel algorithm to jointly optimize resource allocation and video management:* We propose a novel heuristic based on linear programming (LP) termed RAVO. RAVO runs efficiently in polynomial time. It has proven optimality gap on deployment cost and is shown to achieve close-to-optimal decisions on video management and resource allocation.

—*Efficient clustering for large video pool:* To further reduce the computational complexity for large video pool, we propose a fast algorithm based on spectral clustering on the videos. The algorithm achieves significantly reduced runtime complexity (by a factor of $O(|V|)$, where $|V|$ is the number of videos), with little compromise on system cost.

We conduct an extensive simulation and comparison study on RAVO with other state-of-the-art schemes. Our results show that RAVO with spectral clustering achieves substantially the lowest system cost, outperforming the other schemes by a wide margin (often by multiple times). Trace-driven simulation with video data from a large-scale deployed system further confirms our results.

This article is organized as follows. We first review related work in Section 2. We then formulate the joint optimization problem and show that it is NP-hard in Section 3. In Section 4, we present RAVO. We discuss the case of large video pool with our video clustering algorithm in Section 5. We show illustrative simulation results and a trace-driven simulation in Section 6. We conclude and discuss future work in Section 7.

2. RELATED WORK

There has been much work on resource provisioning and rescaling in the cloud for virtual machines and web applications based on online algorithms. Such work often focuses on volatile contents (e.g., short video clips and user generated contents) and assumes that the data centers replicate the full content or the most popular ones without collaboration (we consider a VoD cloud where video cannot be fully and cost-effectively replicated locally). This includes how to predict CPU utilization [Bobroff et al. 2007; Gong et al. 2010; Tang et al. 2007], power consumption [Kusic et al. 2009; Lin et al. 2011; Shuja et al. 2014], service availability [De Melo et al. 2014], network bandwidth utilization [Wang et al. 2011], or efficient storage utilization within a data center [Li et al. 2015; Wang et al. 2015]. There has also been work optimizing server selection and resource provisioning for multimedia cloud [Alasaad et al. 2012; Chu et al. 2014; Niu et al. 2012; Fallah et al. 2015]. All these works differ from ours, and have not considered geo-distributed content replication and retrieval. The cost models they use also fundamentally differ from ours, where we comprehensively consider the cost of server storage, processing, and link bandwidth. We also study the novel joint problem of resource allocation and video management.

Many heuristic approaches have been proposed to optimize VoD resource provisioning in content distribution network (CDN) (in terms of server location, bandwidth requirement, storage, etc.) [Thouin and Coates 2007; Thouin et al. 2006; Laoutaris et al. 2005; Li and Wu 2010]. Another body of VoD work is on optimizing content replication and retrieval in a CDN with fixed server parameters [Borst et al. 2010; Applegate et al. 2010, 2013; Chan and Xu 2013; Hu et al. 2015]. Although various models and optimization goals have been considered, these works have not considered a cloud service where server storage and processing capacities can be rescaled to meet user demand cost-effectively. As CDN is generally regarded as a long-term investment, these works have not considered *jointly* optimizing video management (in terms of server selection and partial content replication) and resource allocation for cooperative servers. By studying such a problem, we can achieve the tradeoff between link bandwidth and server storage/processing, thereof achieving much lower deployment cost with assured service quality. Furthermore, in contrast with previous work, we consider uniquely that video popularity may *differ* for servers at different geographical locations, and hence servers need to replicate videos according to their local demand while collaborating to achieve global cost optimality.

VoD research on peer-to-peer (P2P) assisted streaming usually aims at reducing the load of the repository by effectively using the resource from peers [Yiu et al. 2007; Wu and Lui 2012; Cong et al. 2014; Kurniawan et al. 2014; Lin and Shen 2015; Magharei et al. 2014; Fernando and Keppetiyagama 2013]. They have not considered sufficiently the interaction between system cost and user delay issues. Our work, in contrast, has a different optimization objective of minimizing the deployment cost while ensuring a given user delay requirement.

Table I. Major Symbols Used in Formulation

Notation	Definition
S	The set of servers (central and proxy servers)
V	The set of videos
$L^{(v)}$	Length of video v (seconds)
$p_m^{(v)}$	Access probability of video v at server m
$I_m^{(v)}$	Boolean variable indicating whether server m stores video v
H_m	Storage capacity of server m (bits)
$R_{mn}^{(v)}$	The probability of streaming video v from server m to server n
μ_m	Request rate at server m (requests/second)
$\varepsilon_m^{(v)}L^{(v)}$	Average viewing time of video v at server m (seconds)
$\gamma^{(v)}$	Streaming rate of video v (bits/s)
Γ_{mn}	Average transmission rate from server m to n (bits/s)
K_{mn}	Link capacity from server m to n (bits/s)
U_m	Total upload rate of server m (bits/s)
Λ_m	Processing capacity of server m to handle streaming requests from remote servers (bits/s)
C_{mn}^{N}	Link cost due to directed traffic from server m to n
C_m^{S}	Cost of server m
C	Total deployment cost
D_{mn}^{N}	Delay due to directed traffic from server m to n
D_m^{S}	Delay due to upload streaming of server m
$\bar{D}$	Delay constraint of the system (QoS)

3. PROBLEM FORMULATION AND ITS NP-HARDNESS

In this section, we present the joint optimization problem for video management and resource allocation to minimize deployment cost. We first discuss the modeling of video management (Section 3.1), followed by the user delay constraint (Section 3.2) and resource allocation (Section 3.3), and, finally, the cost optimization problem and its NP-hardness (Section 3.4). We show some of the important symbols in Table I.

The overlay network is modeled as a directed graph $T(S, E)$, where S is the set of central and proxy servers and $E \subseteq S \times S$ is the set of overlay links connecting nodes in S (may not be complete). Let V be the set of videos and $L^{(v)}$ be the video length (in seconds) of v.

3.1. Modeling of Video Management

We model video storage and retrieval in this section. A user is associated with its local server. For cost-effectiveness, each server has a storage capacity to store some videos. The server delivers the locally stored video to the user directly and requests the rest of the video from the remote server.

Let $p_m^{(v)}$ be the popularity of video v at server m, which is the probability that a user requests video v at server m, where $\sum_{v\in V} p_m^{(v)} = 1, \forall m \in S$.

Let $I_m^{(v)}$ be the Boolean variable indicating whether server m stores video v. Obviously, we require

$$I_m^{(v)} \in \{0, 1\}, \quad \forall m \in S, v \in V. \tag{1}$$

Note that for the repository (i.e., central server), we require $I_m^{(v)} = 1, \forall v \in V$.

Server m has a certain storage capacity H_m (bits) to store the video replication. To meet the storage requirement, we require

$$\sum_{v\in V} I_m^{(v)} L^{(v)} \gamma^{(v)} \le H_m, \quad \forall m \in S. \tag{2}$$

If a server does not store the requested video, then it has to retrieve it from other servers. The servers cooperate using the "cache and stream" model, that is, a remote server streams to a user *through* his or her home server. In other words, a home server acts as an intermediate node between the remote server and local users. Let $R_{mn}^{(v)}$ be the probability of supplying video v from server m to server n. We hence must have

$$\sum_{m \in S} R_{mn}^{(v)} = 1, \quad \forall n \in S, v \in V. \tag{3}$$

As the server cannot offer video it does not store, we get

$$0 \leq R_{mn}^{(v)} \leq I_m^{(v)}, \quad \forall m, n \in S, v \in V, \tag{4}$$

and, by definition, $R_{mm}^{(v)} = I_m^{(v)}$.

Let μ_m be the video request rate at server m (requests per second); the request rate for video v at the server m is hence $p_m^{(v)}\mu_m$. Let $\varepsilon_m^{(v)}L^{(v)}$ be the average holding (or viewing) time for video v at server m, where $\varepsilon_m^{(v)} \geq 0$. A user streams the video from the server proportional to his holding time. Therefore, the amount of streamed data from server m to n for video v upon one request is given by $\varepsilon_n^{(v)}R_{mn}^{(v)}L^{(v)}\gamma^{(v)}$. Let Γ_{mn} (bits/s) be the total link bandwidth used for video transmission from server m to n, which can be obtained as

$$\Gamma_{mn} = \sum_{v \in V} p_n^{(v)}\varepsilon_n^{(v)}\mu_n R_{mn}^{(v)}L^{(v)}\gamma^{(v)}, \quad \forall m, n \in S, \tag{5}$$

for $m \neq n$, and, by definition, $\Gamma_{mm} = 0$. The total rate (bits/s) that server m serves other servers is hence

$$R_m = \sum_{n \in S, n \neq m} \Gamma_{mn}, \quad \forall m \in S. \tag{6}$$

Note that we have considered the geographical heterogeneity of user preference and interactivity by allowing $p_m^{(v)}$ and $\varepsilon_m^{(v)}$ for the same video v to vary for each server m (a video v preferred at server m may have $\varepsilon_m^{(v)} > 1$ or vice versa). As we are considering the time-averaged deployment cost at steady state, we are interested in *average* holding time (the distribution of the holding time may vary for different videos). While watching a video, a user holds up a stream and may uniformly request any part of the video over time.

Also note that we have considered the video transcoding issue by allowing the video streaming rates to vary for each video v. A video with multiple streaming rates can be modeled as multiple video versions, and each video version v has its own geo-related popularity $p_m^{(v)}$ and streaming rate $\gamma^{(v)}$.

3.2. Modeling of Delay Constraint

We model the video access delay (user delay) in this section. For good user experience, the waiting time for the video cannot be too long. The total delay of a remote video access mainly consists of the delay on the remote server and the delay due to the network traffic between servers.

For any server $m \in S$, the delay of serving the remote requests (including queueing and processing the requests) depends on the processing capacity of the server (Λ_m) and the total bitrate that server m serves other servers (U_m). Let D_m^{S} be the delay due to the upload streaming of server m to support video requests from remote servers. As more requested bitrate and less processing capacity will cause more delay on server, this delay is a monotonically non-decreasing function in U_m and monotonically non-increasing function in Λ_m, that is,

$$D^{\mathrm{S}}_m = \mathbb{D}^{\mathrm{S}}_m(U_m, \Lambda_m), \quad \forall m \in S. \tag{7}$$

For the network delay between servers, we mainly consider queueing and transmission delay as they play vital roles in ensuring the user experience. Let D^{N}_{mn} be the delay due to the directed traffic from server m to n. It depends on the link bandwidth reserved K_{mn} and the actual data transferred Γ_{mn} through the link. Similarly, as more link traffic and less link bandwidth capacity will cause more delay on link, this delay is a monotonically non-decreasing function in Γ_{mn} and monotonically non-increasing function in K_{mn}, that is,

$$D^{\mathrm{N}}_{mn} = \mathbb{D}^{\mathrm{N}}_{mn}(\Gamma_{mn}, K_{mn}), \quad \forall m, n \in S, \tag{8}$$

with $D^{\mathrm{N}}_{mm} = 0$. Also, D^{N}_{mn} does not have to be the same as D^{N}_{nm}.

To ensure QoS, there is an upper bound $\bar{D}$ of the total delay to retrieve video from other server, which is the sum of the delay due to server and link, that is,

$$D^{\mathrm{N}}_{mn} + D^{\mathrm{S}}_n \leq \bar{D}, \quad \forall m, n \in S. \tag{9}$$

3.3. Modeling of Resource Allocation

We consider a comprehensive cost model for resource allocation in the VoD system. The cost of the VoD system consists of the cost to rent servers in data centers and the cost of link bandwidth capacity between servers.

For any remote server $m \in S$, the cost of a server depends on its total storage, processing capacity (in order to serve the other servers in the network), and actual data transferred. Let C^{S}_m be the cost of server m. Server cost is a monotonically non-decreasing function in H_m, Λ_m, and U_m, that is,

$$C^{\mathrm{S}}_m = \mathbb{C}^{\mathrm{S}}_m(H_m, \Lambda_m, U_m), \quad \forall m \in S. \tag{10}$$

Let C^{N}_{mn} be the link cost due to the directed traffic from server m to n. We pay for the link capacity reserved K_{mn} and the actual data transferred Γ_{mn} through the link. (Usually, we may only be charged by either K_{mn} or Γ_{mn}. In such a case, the price of the other item is 0.) Link cost is a monotonically non-decreasing function in Γ_{mn} and K_{mn}, that is,

$$C^{\mathrm{N}}_{mn} = \mathbb{C}^{\mathrm{N}}_{mn}(\Gamma_{uv}, K_{uv}), \quad \forall m, n \in S, \tag{11}$$

with $C^{\mathrm{N}}_{mm} = 0$. Similarly, C^{N}_{mn} does not have to be the same as C^{N}_{nm}.

Therefore, the total system deployment cost C is

$$C = \sum_{m \in S} C^{\mathrm{S}}_m + \sum_{m,n \in S} C^{\mathrm{N}}_{mn}. \tag{12}$$

Note that, for the generality of the problem formulation, we do not assume the linearity of the delay functions $\mathbb{D}^{\mathrm{S}}_m$, $\mathbb{D}^{\mathrm{N}}_{mn}$ and the cost functions $\mathbb{C}^{\mathrm{S}}_m$ and $\mathbb{C}^{\mathrm{N}}_{mn}$ at this stage. To solve the problem, such functions can all be approximated by piecewise linear functions with good optimality, which is discussed in Section 4.3.

3.4. Cost-optimization Problem and Its NP-hardness

Given the above sections, we state our cost-optimization problem as follows:

Optimal Resource Allocation and Video Management Problem to Minimize Total Deployment Cost (ORVPM): Given topology T, delay upper bound $\bar{D}$, user demand $\{\mu_m\}$, video popularity $\{p^{(v)}_m\}$, cost functions $\{\mathbb{C}^{\mathrm{S}}_m\}$, $\{\mathbb{C}^{\mathrm{N}}_{mn}\}$, and delay function $\{\mathbb{D}^{\mathrm{S}}_m\}$, $\{\mathbb{D}^{\mathrm{N}}_{mn}\}$, our objective function is to minimize the total deployment cost given by Equation (12), that is,

$$\min C = \sum_{m\in S} C_m^{\mathrm{S}} + \sum_{m,n\in S} C_{mn}^{\mathrm{N}}, \tag{13}$$

subject to Equations (1) to (11). The output is the optimal solution of the video stored in each server (that is, $\{I_m^{(v)}\}$), the retrieval probability between servers (i.e., $\{R_{mn}^{(v)}\}$), the link capacity between servers $\{K_{mn}\}$, and the storage and processing capacities of each server (i.e., $\{H_m\}$ and $\{\Lambda_m\}$).

Claim: The joint optimization problem ORVPM is NP-hard.

Proof. We prove its NP-hardness by deriving a polynomial reduction from the Dominating Set Problem, whose NP-complete version is stated as follows. In graph theory, a dominating set for a graph $T = (S, E)$ is a subset $D \subseteq V$ such that every vertex not in D is adjacent to at least one element of D. The domination number $\zeta(T)$ is the number of vertices in a smallest dominating set for T. For a given graph $T = (S, E)$ and input J, is the domination number $\zeta(T)$ at most J?

Given T and J, we construct an instance of our decision problem as follows. The VoD system contains only one video. Consider the case that unit storage cost is 1 per video and unit access cost of "pulling" a video from a remote server (including the cost of server processing capacity and the cost of link capacity and actual data transferred) is 0. Note that such an instance construction can obviously be done in polynomial time. Our decision problem hence becomes: Given this instance, is there a joint optimization strategy which achieves total cost of at most J?

We show that the domination number $\zeta(T)$ is at most J if and only if there is such an optimization strategy. First, if there is such a strategy, any server $m \in S$ either has this video or can access the video through a link $mn \in E$. Then the set of node m whose server has the video forms a dominating set D. The size of this set D is at most J because the storage cost of each server with a video is at least 1 and the total cost is J. On the other hand, if we have a dominating set D with size J, we can easily derive such a strategy by assigning the video to each server in set D. Therefore, we reduce the Dominating Set Problem to our decision problem ORVPM that proves that our optimization problem is NP-hard.

4. RAVO: JOINT OPTIMIZATION FOR RESOURCE ALLOCATION AND VIDEO MANAGEMENT

In this section, we present RAVO, an efficient algorithm which *jointly* optimizes resource allocation and video management. RAVO first relaxes the NP-hard ILP problem given in Section 3 to a continuous linear program (Section 4.1). By using randomized rounding, RAVO accordingly quantizes the LP solution to obtain the video management decisions, that is, the video stored on each server, and the probabilities to retrieve a missing video from remote servers (Section 4.2). Given video management, RAVO then allocates the server storage, server processing and link capacities accordingly (Section 4.3). We finally analyze the complexity (Section 4.4) and optimality of RAVO (Section 4.5) of RAVO.

For clarity, we denote the symbols related to the continuous linear program with superscript hat. For the final result after the randomized rounding by RAVO, we still use the plain symbols.

4.1. Relaxing the Joint Formulation as a Linear Program

To address the NP-hard problem, we first relax the constraint in Equation (1) as $0 \le \hat{I}_m^{(v)} \le 1$ for all $m \in S$ and $v \in V$.

After such relaxation, it is clear that our problem contains only linear constraints of real numbers. For any arbitrary piecewise linear functions of $\mathbb{D}_m^{\mathrm{S}}$, $\mathbb{D}_{mn}^{\mathrm{N}}$, $\mathbb{C}_m^{\mathrm{S}}$, and $\mathbb{C}_{mn}^{\mathrm{N}}$

in Equations (7), (8), (10), and (11), the above problem becomes a linear programming (LP) problem which can be solved efficiently. $\hat{I}_m^{(v)}$ in LP solution refers to the fraction of video v that server m would store.

4.2. Video Management: Storage and Retrieval

We use $\hat{I}_m^{(v)}$ from the joint solution of LP as the probability that video v should be stored at server m. After making the randomized rounding of storage decision on whether to store v at m, we use $I_m^{(v)}$ as the Boolean variable indicating whether server m stores video v.

Let $R_{mn}^{(v)}$ be the probability that server n on a request for video v would retrieve from server m after the video placement described in Section 4.2. We require that $\sum_{m\in S} R_{mn}^{(v)} = 1$.

If server n cannot retrieve video v at any proxy server suggested by the LP solution, that is, $\sum_{m\in S} I_m^{(v)}\hat{R}_{mn}^{(v)} = 0$, then the requests for the video in server n have to be served from the repository. For the other cases, an obvious way to eliminate the probability to retrieve from a server that does not have video v is to assign such probability to other servers m that have video v, and the newly assigned probability is proportional to original $\hat{R}_{mn}^{(v)}$ given by LP. This can be obtained as

$$R_{mn}^{(v)} = \begin{cases} 0, & \text{if } I_m^{(v)} = 0; \\ \dfrac{\hat{R}_{mn}^{(v)}}{\sum\limits_{m\in S} I_m^{(v)}\hat{R}_{mn}^{(v)}}, & \text{if } I_m^{(v)} = 1; \end{cases} \quad \forall m, n \in S. \tag{14}$$

4.3. Resource Allocation: Server Storage, Server Processing and Link Capacities

With $I_m^{(v)}$, the required storage capacity at m is

$$H_m = \sum_{v\in V} I_m^{(v)} L^{(v)} \gamma^{(v)}, \quad \forall m \in S. \tag{15}$$

By Equation (2), it is clear that the expectation value of H_m is $\hat{H}_m$. With a large video pool, we expect that H_m is quite close to $\hat{H}_m$.

With $R_{mn}^{(v)}$, we can calculate Γ_{mn} by

$$\Gamma_{mn} = \sum_{v\in V} p_n^{(v)} \varepsilon_n^{(v)} \mu_n R_{mn}^{(v)} L^{(v)} \gamma^{(v)}, \quad \forall m, n \in S, \tag{16}$$

and U_m by Equation (6).

After getting Γ_{mn} and U_m, we can put them back to function $\mathbb{D}_m^{\mathrm{S}}$ and $\mathbb{D}_{mn}^{\mathrm{N}}$ in Equations (7) and (8) to get the values of Λ_m and K_{mn}. Specifically, we can efficiently solve the piecewise linear equations

$$\mathbb{D}_m^{\mathrm{S}}(U_m, \Lambda_m) = D_m^{\mathrm{S}}, \quad \forall m \in S, \tag{17}$$

and

$$\mathbb{D}_{mn}^{\mathrm{N}}(\Gamma_{mn}, K_{mn}) = D_{mn}^{\mathrm{N}}, \quad \forall m, n \in S, \tag{18}$$

where D_m^{S} and D_{mn}^{N} are from the LP solution. Clearly, the QoS constraints described in Equation (9) are satisfied.

4.4. Algorithmic Complexity

To solve the LP, we may employ the wide-region centering-predictor-corrector algorithm (an interior-point method). The number of variables of the formulation is $O(|S|^2|V|)$.

Therefore, it has a $O(|S||V|^{1/2})$ worst-case iteration bound and $O(|S|^7|V|^{3.5})$ overall worst-case time complexity. Usually, the iteration number is constant and the problem is expected to be solved in $O(|S|^6|V|^3)$ time.

Furthermore, the time complexity to decide video storage ($\bar{I}_m^{(v)}$) is $O(|S||V|)$, because all the servers and videos are traversed to determine which server to store which video. Similarly, to decide video retrieval ($\bar{R}_{mn}^{(v)}$), we have to traverse all the links and videos. Therefore, the complexity is $O(|S|^2|V|)$.

Given the above, the overall time complexity of RAVO is

$$O(|S|^6|V|^3 + |S|^2|V|). \tag{19}$$

Note that for large $|V|$, the term $O(|S|^6|V|^3)$ will dominate.

4.5. Storage and Traffic Optimality

The cost is due to either the server storage or the traffic between servers. If we can prove the optimality of these two components, then the optimality of RAVO is guaranteed.

We show that the expected value of the storage allocated for each server is the same as the value of the LP solution. We also show that after randomized rounding, the expected value of extra traffic to the repository is less than $1/e$ of the total traffic.

Claim: The expected value of storage after the quantization $E(H_m)$ is same as the LP solution $\hat{H}_m$.

Proof. By Equation (15) we get

$$\begin{aligned} E(H_m) = E\left(\sum_{v\in V} I_m^{(v)} L^{(v)} \gamma^{(v)}\right) &= \sum_{v\in V} E\big(I_m^{(v)}\big) L^{(v)} \gamma^{(v)} \\ = \sum_{v\in V} \hat{I}_m^{(v)} L^{(v)} \gamma^{(v)} &= H_m, \quad \forall m \in S. \end{aligned} \tag{20}$$

Therefore, it is expected that there is no extra cost for storage due to the quantization.

Claim: The expected value of extra traffic to the repository is less than $1/e$ of the total traffic.

Proof. As mentioned in Section 4.2, if a server cannot retrieve video from any proxy server suggested by the LP solution, the request will be served from the repository. This causes extra cost for network traffic and server streaming. We show that such a probability is less than $1/e$. We require that all the video can be retrieved from the proxies in the network, therefore $\sum_m \hat{I}_m^{(v)} \geq 1$ for the relaxed linear program. The probability that none of the proxy server has the video is $\prod_m (1 - \hat{I}_m^{(v)})$, which is maximized when $\hat{I}_m^{(v)} = 1/|S|\ \forall m \in S$, where $|S|$ is the number of servers. So the upper bound of the probability can be written as $(1 - 1/|S|)^{|S|}$, which is bounded by $1/e$. (i.e., $\prod_m (1 - \hat{I}_m^{(v)}) \leq (1 - 1/|S|)^{|S|} \leq 1/e$.)

The cost due such extra traffic also depends on the price for the repository's processing power and the link price to the repository. For a practical setting such as when the price of serving streaming at the repository is twice as serving it in the original proxy server, the extra cost caused by quantization (i.e. the optimality gap) is less than $1/e$ of the original cost.

Note that since the time complexity of randomized rounding is quite low, it is possible that we run the randomized rounding algorithm multiple times and choose the best result to avoid the worst case.

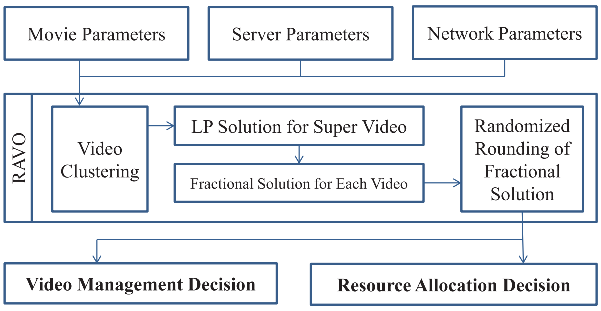
Movie Parameters
Server Parameters
Network Parameters
RAVO
Video Clustering
LP Solution for Super Video
Fractional Solution for Each Video
Randomized Rounding of Fractional Solution
Video Management Decision
Resource Allocation Decision

Fig. 2. Framework of RAVO with video clustering.

5. EFFICIENT COMPUTATION FOR LARGE VIDEO POOL

Despite its efficiency, the running time of RAVO is still high for large video pools (note the factor $O(|V|^3)$ in Equation (19)). To further reduce the runtime complexity, we propose an efficient video clustering algorithm for a large video pool based on spectral clustering in Section 5.1. Video management and resource allocation are then allocated (Section 5.2). We analyze its algorithmic reduction in Section 5.3.

Video clustering with k-means formulation has been used in Chang and Chan [2015], which assumes uniform streaming rate and video popularity among servers so it can be solved in polynomial time. This method cannot be directly used here due to heterogeneous streaming rate and geo-heterogeneity of the video popularity. Generally speaking, most traditional clustering methods (such as k-means, k-medians, k-medoids, and hierarchical clustering) are NP-hard when we have geo-heterogeneity, and therefore they cannot reduce the time complexity of RAVO with a large video pool. Our approach used in this section is based on spectral clustering [Fowlkes et al. 2004] and has a clear polynomial time complexity bound even with large video pool and geo-heterogeneity, which can effectively address these issues.

We show the approach to implement RAVO with spectral clustering in Figure 2. Given the system parameters on the movies, servers, and network, the server first cluster the videos into “super videos” to reduce the problem size as described in Section 5.1. Then the server can solve the relaxed LP problem and get the continuous solution for the “super videos” as presented in Section 4.1. With this solution, the server further calculates the continuous solution for each video as presented in Section 5.2. Based on the algorithm in Sections 4.2 and 4.3, the continuous solution is quantized. The video management decisions for each video and resource allocation decisions for each server and link can then be made accordingly.

5.1. Efficient Video Clustering Based on the Metric of Concurrency Density

In order to cluster the videos into groups, we shall consider the storage and streaming traffic as they both contribute to the total deployment cost. To describe the streaming traffic, we consider the user concurrency, which is the average number of users to request the video at one time. The *concurrency density* is defined as the user concurrency divided by the storage requirement, which gives the per-storage user concurrency of a video. The key idea of clustering is to put the videos with similar concurrency densities into the same group by minimizing the sum of the difference within each group.

To formulate the problem, we begin by letting $b_m^{(v)}$ be the *concurrency density* for video v at server m, which can be written as

$$b_m^{(v)} = p_m^{(v)} \varepsilon_m^{(m)}, \quad \forall m \in S, v \in V. \tag{21}$$

Let G be the set of video groups and g_i be the ith group in G, and the number of groups $|G|$ is given as a system parameter. To further motivate our approach, consider a set of videos v in group g. From Equation (5), the link transmission due to videos in g is

$$\Gamma_{mn}^{(g)} = \sum_{v \in g} \mu_n b_n^{(v)} R_{mn}^{(v)} L^{(v)} \gamma^{(v)}, \quad \forall m, n \in S. \tag{22}$$

If all the videos $v \in g$ has the same $b_n^{(v)}$ (given by $b_n^{(g)}$) and $R_{mn}^{(v)}$ (given by $R_{mn}^{(g)}$), then Equation (22) can be written as

$$\Gamma_{mn}^{(g)} = \mu_n b_n^{(g)} R_{mn}^{(g)} \sum_{v \in g} L^{(v)} \gamma^{(v)}, \quad \forall m, n \in S. \tag{23}$$

From above, it is clear that if we group the videos with the same $b_n^{(v)}$ and access the video with the same $R_{mn}^{(v)}$, the videos in g can be regarded as an aggregated "super video" with concurrency density $b_n^{(g)}$ and file size $L^{(g)}\gamma^{(g)} = \sum_{v \in g} L^{(v)}\gamma^{(v)}$ for the linear program.

In the case that the videos are of different $b_m^{(v)}$, we cluster the videos with similar $b_m^{(v)}$ into one group and minimize the sum of the distances of concurrency densities within each group. Denote

$$\boldsymbol{b}^{(v)} = \left(b_1^{(v)}, b_2^{(v)}, \ldots, b_{|S|}^{(v)}\right) \tag{24}$$

as a $|S|$ dimensional vector where each dimension represent the popularity of video v in a server. The distance between two video popularities can be measured as distance between two vectors (e.g., Euclidean distance). Therefore, the objective of our video grouping algorithm is to minimize

$$\arg_{g_i} \sum_{i=1}^{|G|} \sum_{v \in g_i} \left\| \boldsymbol{d}^{(v)} - \tilde{\boldsymbol{d}}^{(g_i)} \right\|^2, \tag{25}$$

where $\tilde{\boldsymbol{d}}^{(g_i)}$ is the mean $\boldsymbol{d}^{(v)}$ of group g_i. This formulation is exactly k-means, a method to partition data into clusters in which each datum belongs to the cluster with the nearest mean. Note that the group size of each group may not be the same by k-means clustering.

The above grouping scheme leads to the following:

—*Length*: Each group size $L^{(g_i)}$ is given by

$$L^{(g_i)} = \sum_{v \in g_i} L^{(v)}, \quad \forall g_i \in G. \tag{26}$$

—*Average streaming rate*: The average streaming rate of each group $\gamma^{(g_i)}$ is given by

$$\gamma^{(g_i)} = \frac{\sum_{v \in g_i} L^{(v)} \gamma^{(v)}}{\sum_{v \in g_i} L^{(v)}}, \quad \forall g_i \in G. \tag{27}$$

—*Group concurrency density*: The group concurrency density $b_m^{(g_i)}$ is given by

$$b_m^{(g_i)} = \frac{\sum_{v \in g_i} b_m^{(v)} L^{(v)} \gamma^{(v)}}{L^{(g_i)} \gamma^{(g_i)}}, \quad \forall g_i \in G, m \in S. \tag{28}$$

After video grouping, we run linear programming on these groups by treating them as $|G|$ "super videos" with concurrency density $\boldsymbol{b}^{(g_i)}$, length $L^{(g_i)}$, and streaming rate $\gamma^{(g_i)}$.

5.2. Video Management and Resource Allocation

We discuss in this section how to obtain the quantized parameters for each video v. In server m, we have $I_m^{(g_i)} L^{(g_i)} \gamma^{(g_i)}$ space to store all the videos $v \in g_i$. All the videos in the same group should have similar number of replicas in the cloud. Therefore, we use *rarest first* in video placement. When a server makes a placement decision for a group g_i, it selects the video v in group g_i that is the least globally stored until the space budget of the group g_i is fully consumed.

In our retrieval algorithm, we can make $R_{mn}^{(v)} = R_{mn}^{(g_i)}$ for $v \in g_i$. Then we use the method in Section 4 for further parameter quantization.

5.3. Complexity Reduction

General k-means clustering problem is NP-hard and the algorithmic complexity of its solutions is not easy to analyze. The recent development of spectral clustering has better performance and a clearer time complexity of $O(|S||V|)$ [Fowlkes et al. 2004].

After we group the videos, the complexity to solve the linear program is reduced to $O(|S|^6|G|^3)$. Since the quantization takes $O(|S|^2|V|)$ time, the total complexity is $O(|S|^6|G|^3 + |S|^2|V|)$. In terms of $|V|$, the complexity is reduced to linear in $|V|$ (from $O(|V|^3)$. In addition, for the solution of linear programming, the complexity is reduced from $O(|S|^6|V|^3)$ to $O(|S|^6|G|^3)$, which is very substantial (e.g., $|G|$ is around hundreds, while $|V|$ can be more than tens of thousands).

Note that we can adjust $|G|$ to make the tradeoff between time complexity and optimality. For a larger $|G|$, we can achieve better optimality at the cost of higher time complexity.

6. ILLUSTRATIVE SIMULATION RESULTS

In this section, we first present our simulation environment and performance metrics in Section 6.1, followed by illustrative results in Section 6.2. To further validate our work, we give trace-driven simulation results based on real data from a large-scale deployed VoD system [Liu et al. 2014] in Section 6.3.

6.1. Simulation Environment and Performance Metrics

RAVO is general enough to be applied to any pricing scheme, any level of video popularity, and level of geographically heterogeneous video popularity. For concreteness in our simulation, we consider that the local video popularity follows the Zipf distribution with Zipf parameter z, that is, the request probability of the ith video, denoted as $f(i)$, is given by $f(i) \propto 1/i^z$. To address the geographic heterogeneity of the popularity, we allow a fraction of the video rank to differ for different servers. Denote the heterogeneity coefficient ρ $(0 \leq \rho \leq 1)$ to be the fraction of videos to be reshuffled. For each server, we randomly pick up $\rho|V|$ from $|V|$ videos and randomly insert them back. Note that this process only changes the order of videos, but the Zipf distribution remains the same.

In our simulation, we consider that requests arrive at each proxy server according to a Poisson process with total rate μ (req./second). From Section 4, it is clear that RAVO does not depend on the specific request process or trace pattern so long as the request rate is same.

To address RAVO's capability of utilizing cheap resources, we consider that the price of link bandwidth for each link follows a Zipf distribution (independent of each other).

Table II. Baseline Parameters Used in Our Study

Parameter	Default value
Number of proxy servers	20
Number of videos	10,000
Number of video groups	400
Server storage price σ_H	0.015
Server processing capacity price c_m	0.005
Zipf parameter of local video popularity	1.2
Video length	90 minutes
Average video holding time	Video length ($\varepsilon^{(m)} = 1$)
Video streaming rate	1 Mbits/s
Total request rate in the network	30 req./s (equally distributed to proxies)
Link price between central and proxy server c_{mn}	0.05
Link price between proxies c_{mn}	Zipf with parameter 0.6 and mean 0.005
Heterogeneity coefficient	0.3
Delay constraint $\bar{D}$	1

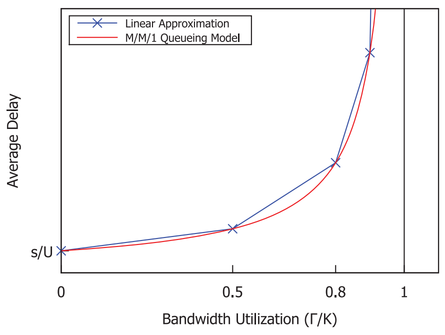

Fig. 3. Delay model for a link between servers.

The VoD cloud consists of a number of distributed proxy servers. All our results are obtained at steady state. Unless otherwise stated, we use the default values shown in Table II for our system parameters (the baseline case).

We use M/M/1 queueing model to calculate the delay in each server and link. We show in Figure 3 the link delay D^{N}_{mn} versus Γ_{mn}/K_{mn} in our simulation, where K_{mn} is the link capacity from server m to server n and hence Γ_{mn}/K_{mn} is the link utilization of the link. A piecewise linear function with 4 linear segments approximates the M/M/1 delay curve. The delay increases more sharply with the link utilization as the consumed bandwidth Γ_{mn} approaches the link capacity K_{mn}. The delay caused by server processing D^{S}_{m} depends on U_m/Λ_m and has the similar curve.

We consider the link cost from server m to server n proportional to the link capacity between them, that is,

$$C^{\mathrm{N}}_{mn} = \mathbb{C}^{\mathrm{N}}_{mn}(\Gamma_{uv}, K_{uv}) = c_{mn}K_{mn}, \;\; \forall m, n \in S, \tag{29}$$

where c_{mn} is some constant (by definition, $c_{mm} = 0$).

The cost of a server is a function of its storage and its total processing capacity to serve the remote servers, modeled as

$$C^{\mathrm{S}}_{m} = \mathbb{C}^{\mathrm{S}}_{m}(H_m, \Lambda_m, U_m) = \sigma_H H_m + c_m\Lambda_m, \;\; \forall m \in S. \tag{30}$$

To validate our simulation settings, we show in Table III the price from Google Cloud Platform [Google 2015], which indicates that linear functions fit the price of link cost

Table III. Link Cost (per GB) from Google Cloud

Used monthly	To China	To Australia	To Others
0-1 TB	$ 0.23	$ 0.19	$ 0.12
1-10 TB	$ 0.22	$ 0.18	$ 0.11
10+ TB	$ 0.20	$ 0.15	$ 0.08

well. Google [2015] also indicates the storage cost as fixed at $0.026 for a GB per month, which agrees with our storage price.

The performance metrics we are interested in are as follows:

—*Total cost*, which is the sum of server cost and link cost according to Equation (12). This is the deployment cost of the network.
—*Server cost*, which is the sum of its storage and processing capacities defined in Equations (10) and (30). We further examine the following cost components: *Storage cost* is the total cost due to server storage. *Processing cost* is the cost of server processing capacity to support other servers.
—*Link cost*, which is the cost due to traffic between servers defined in Equations (11) and (29).
—*Delay*, which is the maximum delay caused by links and servers.

As mentioned in Section 2, previous work seldom considers the joint optimization of resource allocation and content management, while simply combining an existing resource allocation scheme with a content management scheme cannot assure the service quality. To cover all the important aspects, we extend some state-of-the-art work as the comparison schemes. We compare RAVO with the following schemes:

—*iGreedy* [Laoutaris et al. 2005] with optimal resource allocation, where each proxy server stores the locally most popular videos and retrieves other videos from the repository. This scheme considers the local video popularity but does not take advantage of cooperative replication. It optimizes between the server storage and processing capacities to achieve low cost for the resource allocation.
—*IPTV (Internet Protocol television) Resource Allocation and Management (IPTV-RAM)* [Li and Wu 2010] with optimal content management, which divides the videos into two categories: those globally popular ones that all servers store (full replication) and those globally unpopular ones that only some proxy servers store. By selecting server locations for unpopular videos, it seeks to minimize deployment cost.
—*Super-optimum*, which is the LP solution without quantization. We call this the *super-optimum* solution because it is no worse than the NP-hard *real optimal* solution. Note that RAVO must be no better than the NP-hard solution. We will see in Section 6.2 that RAVO is very close to the *Super-optimum*. Therefore, RAVO is also close to the NP-hard solution.

6.2. Illustrative Results

We compare in Figure 4 the total deployment cost versus the total user request rate for different schemes. Total cost increases with request rate due to the increase in link traffic and storage. RAVO clearly achieves much lower total cost compared with iGreedy and IPTV-RAM, beating it by a large margin. In other words, given the same deployment budget, RAVO can support a much higher request rate (i.e., more concurrent users in the system). iGreedy does not perform well because it mainly relies on the central server to serve the requests for the unpopular videos. IPTV-RAM uses global popularity instead of considering the geographic heterogeneity, assuming that a video has the same popularity at all the servers. Therefore, IPTV-RAM may not store the most locally popular video in the proxy server, which increases the cost.

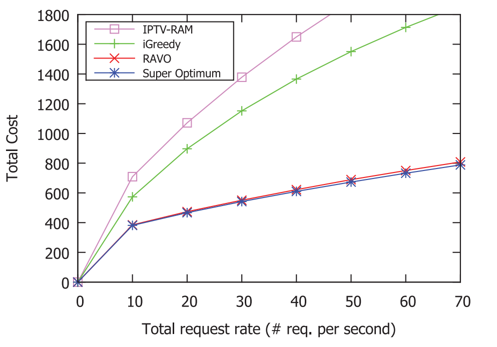

Fig. 4. Deployment cost given different request rate.

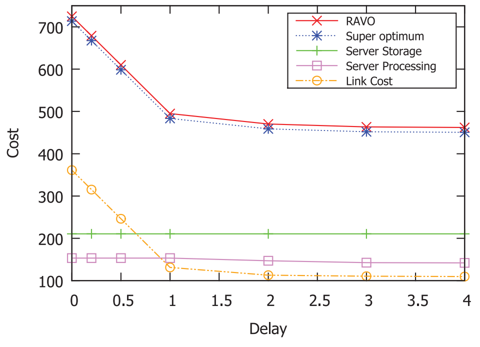

Fig. 5. Deployment and component cost given different delay requirement.

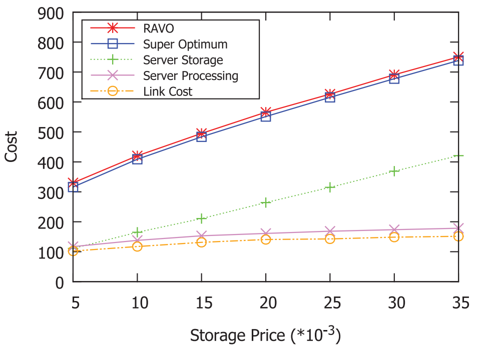

Fig. 6. Total deployment and component cost given difference storage price.

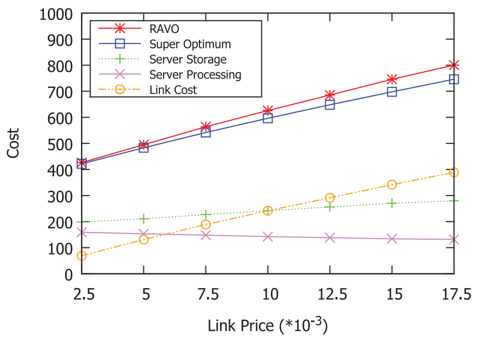

Fig. 7. Total deployment and component cost given difference link price.

We compare in Figure 5 the total cost versus the delay constraint for different components of the deployment cost. Total cost increases with a more demanding delay constraint. RAVO clearly achieves close-to-optimum performance as the gap between the RAVO and Super-optimum is quite small. Server storage and sever processing cost remain nearly unchanged but the link cost increases drastically with the demand of the lower delay constraint. This indicates that increasing link capacity is more effective to reduce the total delay.

We show in Figure 6 the cost components and total cost versus storage price. The server storage cost increases with the increasing of the storage price, but the cost of server processing and link capacity remains steady. This indicates that a minimum requirement of storage is needed to keep the popular videos.

We show in Figure 7 the cost components and total cost versus link price. The storage cost and the link cost both increase with the increasing of the link price, but the processing cost decreases. This shows the tradeoff between storage and processing. When the link price is higher, it is economical to store videos locally.

We plot in Figure 8 the individual server cost for different schemes. We sort the proxy servers according to the average link price to retrieve video from a server in descending order, and the last one refers to the repository. With cheap link price, server cost increases because it will serve more remote requests. RAVO utilizes very well the cheap resources of proxy servers, leading to significantly lower server cost.

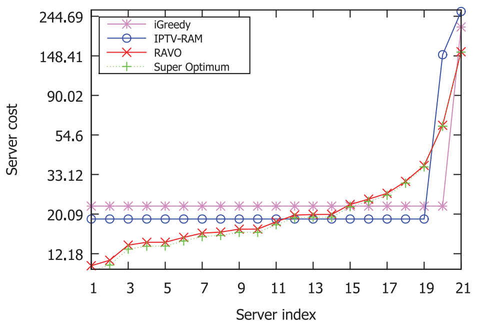

Fig. 8. Server cost distribution given different schemes.

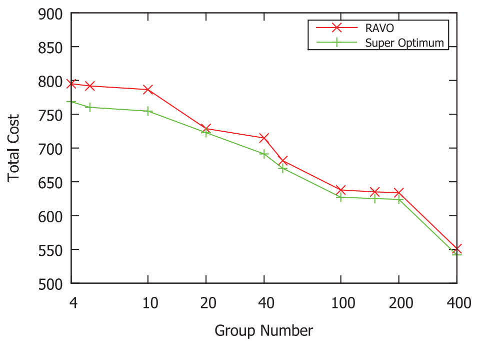

Fig. 9. Total cost versus group number.

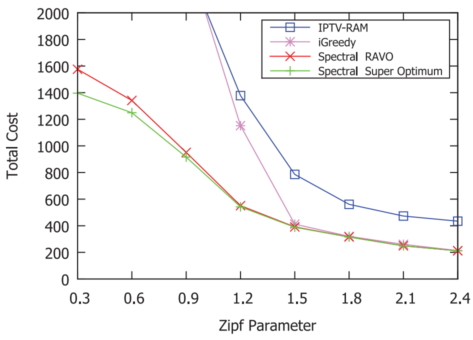

Fig. 10. Deployment cost given different Zipf parameter of video popularity.

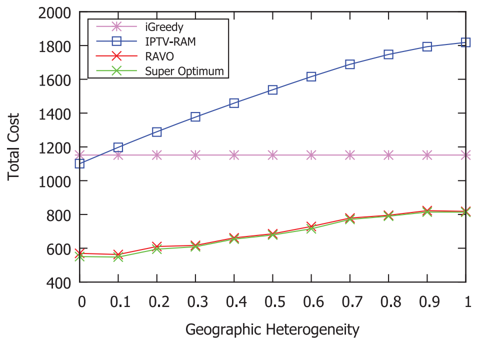

Fig. 11. Deployment cost given different heterogeneity of video popularity.

As iGreedy only stores the most popular videos at the proxy servers, it has lower proxy cost but high repository cost. IPTV-RAM chooses a cheap proxy server for unpopular videos, but the popularity heterogeneity makes fewer videos globally popular. Therefore, the storage and processing cost for globally unpopular videos is even higher.

We plot in Figure 9 the total cost versus group number for video clustering. Due to the large video number, it is impossible to calculate the unclustered super-optimum. As group number increases, the load indices in the same group are closer to each other and the problem is better approximated. Therefore, the cost decreases with more groups (and more computation time).

We plot in Figure 10 the total cost versus the Zipf parameter of video popularity given different schemes. The total cost in general decreases with the skewness. This is because more requests are concentrated on fewer popular videos, which leads to lower miss rate and hence lower processing and link cost. In other words, iGreedy and IPTV-RAM perform better with higher Zipf parameter because the decision made from RAVO, iGreedy, and IPTV-RAM become more similar. However, RAVO achieves the lowest total cost even for low skewness (i.e. when the popularity is quite uniform). This shows that RAVO makes good decision on video management and resource allocation.

We plot in Figure 11 the total cost versus the geographic heterogeneity of video popularity given different schemes. The total cost of RAVO increases with the geographic heterogeneity. This is because higher geographic heterogeneity will introduce more error in the clustering process. The cost of iGreedy is unchanged because the Zipf

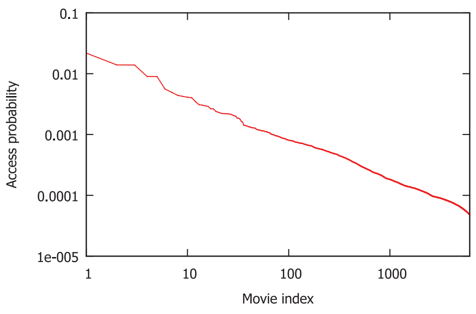

Fig. 12. Movie access probability in descending order.

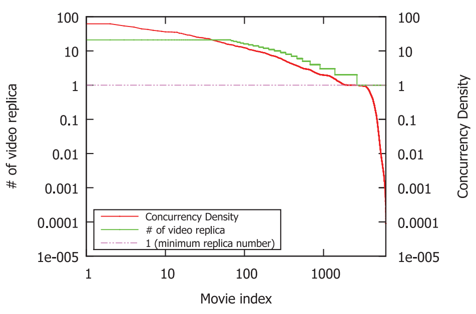

Fig. 13. Concurrency density and replica number versus movie index.

parameter of local popularity is the same. IPTV-RAM, on the other hand, has a much higher cost with more heterogeneous popularity because the difference between global and local popularity is larger. Still, RAVO is close to the super optimum and outperforms iGreedy and IPTV-RAM by a wide margin.

6.3. Trace-Driven Simulation Results

To further validate our work, we give trace-driven simulation. The original data about time-varied traffic, movie traffic proportion, and file size are from a major IPTV service provider company of China, with traces covering about 2 million users over nearly 5 months [Liu et al. 2014]. The traces include user access logs, composed of movie id, viewing time, movie file size, bit rate, and so on. We select 6,148 Netflix-like videos where each video's length is over 40 minutes (i.e., 2,400s). The simulation is carried out based on 5-day trace data (i.e., a planning phase). For other simulation settings unrelated to the trace data, we still use the baseline parameters.

We plot in Figure 12 the movie access probability in descending order. In log-log scale, the plot is nearly linear, which demonstrates that the movie access probability follows Zipf's distribution. By regression, the Zipf parameter for the trace data is 0.80. The results agree with our simulation assumptions in the previous subsection.

We plot in Figure 13 the concurrency density and replica number versus movie index. Note that in Figures 12 and 13 the movie indices differ, as in Figure 13 we sort the movies in descending order according to their concurrency densities. The concurrency density follows a distribution different from the movie access probability, where for movies with concurrency density less than 1 the skewness is drastically increased. Usually a movie with concurrency density less than 1 indicates that most of the users do not finish watching the whole movie. In Figure 13, we also see that a video with higher concurrency density will have more replicas on the cloud. This is because such movies have more user demands so it is more cost-efficient to have more copies over the proxies.

As from the trace data, the level of user demand is fixed. To show how the deployment cost changes with more user requests, we add some synthetic trace data to increase the user requests rate. The newly added synthetic requests have the same statistical features of the original real-world data.

We show in Figure 14 the total deployment cost versus the total user request rate for different schemes. The results further confirm the results in Figure 4. With real-world trace data, RAVO still outperforms the comparison schemes by a large margin. In Figure 15, we show the component cost of RAVO. The storage cost increases more slowly than the other two components when the request rate increases. This is due to the constraint on the replica number. Each movie can at most have one copy per proxy

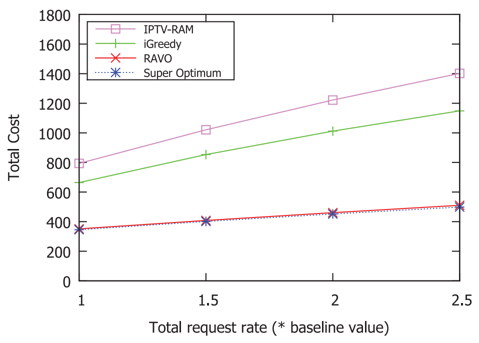

Fig. 14. Deployment cost given different request rate.

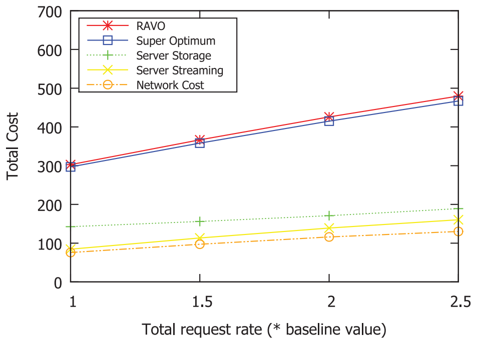

Fig. 15. Deployment and component cost given different request rate.

and at least one copy in the repository. Therefore, there is an upper and a lower limit for each movie's storage cost. Also, as shown in Figure 13, there are a lot of movies where the user even does not want to finish watching them. Such movies' storage cost is insensitive to the change of user requests. In other word, one copy in the repository is good enough for them.

7. CONCLUSION AND FUTURE WORK

We have studied the novel problem of jointly optimizing video management and resource allocation for a large-scale VoD cloud with distributed collaborative servers where videos have geographically heterogeneous popularity. The deployment cost consists of renting link capacities, server storage, and server processing capacity to serve the misses at remote servers. We seek to minimize the deployment cost given a certain QoS constraint on user delay.

We first formulate the joint optimization problem. Our model captures the important system parameters. After proving the problem NP hard, we propose RAVO, a novel algorithm to jointly allocating system resources and managing videos to achieve low deployment cost. For a large video pool, we further present a video clustering algorithm that achieves close-to-optimal performance with a substantial (polynomial) reduction in running time.

We have conducted extensive simulation to compare the performance of RAVO with other state-of-the-art schemes. Our results show that RAVO significantly outperforms the other schemes with much lower deployment cost (cutting it by multiple times). Our trace-driven simulation based on real data further confirms its performance.

There is some future work to be done. First, cloud data centers now provide autoscaling feature so one can rescale a resource, such as storage and streaming capacity, within minutes. It is an interesting problem to utilize this feature to accommodate the variation of user request over a day and further reduce the deployment cost. Also, we can integrate the current VoD network architecture with fog computing so users can contribute their own resource for efficient video delivery.

REFERENCES

Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang. 2012. Unreeling netflix: Understanding and improving multi-CDN movie delivery. In *Procceedings of IEEE INFOCOM 2012*. IEEE, 1620–1628.

Vaneet Aggarwal, Xu Chen, Vijay Gopalakrishnan, Rittwik Jana, K. K. Ramakrishnan, and Vinay A. Vaishampayan. 2011. Exploiting virtualization for delivering cloud-based IPTV services. In *Proceedings* *of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, *Shanghai, China*. IEEE, 637–641.

Amr Alasaad, Kaveh Shafiee, Sathish Gopalakrishnan, and Victor Leung. 2012. Prediction-based resource allocation in clouds for media streaming applications. In *Proceedings of the IEEE Globecom Workshops, Anaheim, CA*. IEEE, 753–757.

David Applegate, Aaron Archer, Vijay Gopalakrishnan, Seungjoon Lee, and K. K. Ramakrishnan. 2013. Content placement via the exponential potential function method. In *Integer Programming and Combinatorial Optimization*. Springer, New York, NY.

David Applegate, Aaron Archer, Vijay Gopalakrishnan, Seungjoon Lee, and Kadangode K. Ramakrishnan. 2010. Optimal content placement for a large-scale VoD system. In *Proceedings of the 6th International Conference (Co-NEXT'10)*. ACM, Philadelphia, PA, 4.

Norman Bobroff, Andrzej Kochut, and Kirk Beaty. 2007. Dynamic placement of virtual machines for managing sla violations. In *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 119–128.

Sem Borst, Varun Gupta, and Anwar Walid. 2010. Distributed caching algorithms for content distribution networks. In *Proc. of the IEEE INFOCOM*. IEEE, 1–9.

S.-H. Gary Chan and Zhuolin Fannie Xu. 2013. LP-SR: Approaching optimal storage and retrieval for video-on-demand. *IEEE Trans. Multimed.* 15, 8 (Dec. 2013), 2125–2136.

Zhangyu Chang and S.-H. Gary Chan. 2015. Bucket-filling: An asymptotically optimal VoD network with source coding. *IEEE Trans. Multimed.* 17, 5 (May 2015), 723–735.

Y.-M. Chu, N.-F. Huang, and S.-H. Lin. 2014. Quality of service provision in cloud-based storage system for multimedia delivery. *IEEE Syst. J.* 8, 1 (Mar. 2014), 292–303.

Xin Cong, Kai Shuang, Sen Su, and FangChun Yang. 2014. An efficient server bandwidth costs decreased mechanism towards mobile devices in cloud-assisted P2P-VoD system. *Peer-to-Peer Network. Appl.* 7, 2 (2014), 175–187.

Rosangela Maria De Melo, Maria Clara Bezerra, Jamilson Dantas, Rubens Matos, Ivanildo José De Melo Filho, and P. M.. 2014. Redundant VoD streaming service in a private cloud: Availability modeling and sensitivity analysis. *Math. Probl. Eng.* 1, 1 (2014), 1–14.

Monireh Fallah, Mostafa Ghobaei Arani, and Mehrdad Maeen. 2015. NASLA: Novel auto scaling approach based on learning automata for web application in cloud computing environment. *Int. J. Comput. Appl.* 117, 2 (2015), 18–23.

Tharidu Fernando and Chamath Keppetiyagama. 2013. ISP friendly peer selection in bittorrent. In *Proceedings of the 2013 IEEE International Conference on Advances in ICT for Emerging Regions (ICTer)*. IEEE, 160–167.

Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. 2004. Spectral grouping using the Nystrom method. *IEEE Trans. Pattern Anal. Mach. Intell.* 26, 2 (2004), 214–225.

Zhenhuan Gong, Xiaohui Gu, and John Wilkes. 2010. Press: Predictive elastic resource scaling for cloud systems. In *Proceedings of the IEEE International Conference on Network and Service Management (CNSM)*. IEEE, 9–16.

Google. 2015. Google Cloud Platform. Retrieved April 1, 2015, from https://cloud.google.com/pricing/.

H. Hu, Y. Wen, T. Chua, J. Huang, W. Zhu, and X. Li. 2015. Joint content replication and request routing for social video distribution over cloud CDN: A community clustering method. *IEEE Trans. Circ. Syst. Vid. Technol.* PP, 99 (2015), 1–1.

Cisco Visual Networking Index. 2013. Forecast and methodology, 2014–2019. *White Paper* 1, 1 (2013), 1–14.

Ikhsan Putra Kurniawan, Hidayat Febiansyah, and Jin Baek Kwon. 2014. Cost-effective content delivery networks using clouds and nano data centers. In *Ubiquitous Information Technologies and Applications*. Springer, New York, NY.

Dara Kusic, Jeffrey O. Kephart, James E. Hanson, Nagarajan Kandasamy, and Guofei Jiang. 2009. Power and performance management of virtualized computing environments via lookahead control. *Cluster Comput.* 12, 1 (2009), 1–15.

Nikolaos Laoutaris, Vassilios Zissimopoulos, and Ioannis Stavrakakis. 2005. On the optimization of storage capacity allocation for content distribution. *Comput. Netw.* 47, 3 (2005), 409–428.

He Li, Mianxiong Dong, Xiaofei Liao, and Hai Jin. 2015. Deduplication-based energy efficient storage system in cloud environment. *Comput. J.* 58, 6 (2015), 1373–1383.

Haitao Li, Lili Zhong, Jiangchuan Liu, Bo Li, and Ke Xu. 2011. Cost-effective partial migration of VoD services to content clouds. In *Proceedings of the IEEE International Conference on Cloud Computing (CLOUD)*, Washington DC. IEEE, 203–210.

Mingfu Li and Chun-Huei Wu. 2010. A cost-effective resource allocation and management scheme for content networks supporting IPTV services. *Comput. Commun.* 33, 1 (2010), 83–91.

Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. 2011. Dynamic right-sizing for power-proportional data centers. In *Proceedings of the 2011 IEEE INFOCOM*. IEEE, 1098–1106.

Yuhua Lin and Haiying Shen. 2015. Autotune: Game-based adaptive bitrate streaming in P2P-assisted cloud-based vod systems. In *Proceedings of the 2015 IEEE International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 1–10.

Ning Liu, Huajie Cui, S.-H. Gary Chan, Zhipeng Chen, and Yirong Zhuang. 2014. Dissecting user behaviors for a simultaneous live and VoD IPTV system. *ACM Trans. Multimed. Comput. Commun. Appl.* 10, 3 (Apr. 2014), 23:1–23:16.

Nazanin Magharei, Reza Rejaie, Ivica Rimac, Volker Hilt, and Markus Hofmann. 2014. ISP-friendly live P2P streaming. *IEEE/ACM Trans. Netw.* 22, 1 (2014), 244–256.

Di Niu, Hong Xu, Baochun Li, and Shuqiao Zhao. 2012. Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In *Proceedings of the IEEE INFOCOM*, IEEE, 460–468.

Junaid Shuja, Kashif Bilal, Sajjad Ahmad Madani, and Samee U. Khan. 2014. Data center energy efficient resource scheduling. *Cluster Comput.* 17, 4 (2014), 1265–1277.

Chunqiang Tang, Malgorzata Steinder, Michael Spreitzer, and Giovanni Pacifici. 2007. A scalable application placement controller for enterprise data centers. In *Proceedings of the 16th International Conference on World Wide Web*. ACM, New York, NY, 331–340.

Frederic Thouin and Mark Coates. 2007. Video-on-demand server selection and placement. In *Managing Traffic Performance in Converged Networks*. Springer, New York, NY.

Frederic Thouin, Mark Coates, and Dominic Goodwill. 2006. Video-on-demand equipment allocation. In *Proceedings of the 5th IEEE International Symposium on Network Computing and Applications*, *Cambridge, MA*. IEEE, 103–110.

Fuguang Wang, Zhuzhong Qian, Sheng Zhang, Mianxiong Dong, and Sanglu Lu. 2015. SmartRep: Reducing flow completion times with minimal replication in data centers. In *Proceedings of the 2015 IEEE International Conference on Communications (ICC)*, IEEE, 460–465.

Meng Wang, Xiaoqiao Meng, and Li Zhang. 2011. Consolidating virtual machines with dynamic bandwidth demand in data centers. In *Proceedings of the IEEE 2011INFOCOM, Shanghai, China*. IEEE, 71–75.

Weijie Wu and John Lui. 2012. Exploring the optimal replication strategy in P2P-VoD systems: Characterization and evaluation. *Trans. Parallel Distrib. Syst.* 23, 8 (2012), 1492–1503.

Yu Wu, Chuan Wu, Bo Li, Xuanjia Qiu, and Francis C. M. Lau. 2011. Cloudmedia: When cloud on demand meets video on demand. In *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS)*, Minneapolis, MMN. IEEE, 268–277.

Wai-Pun Ken Yiu, Xing Jin, and S.-H. Gary Chan. 2007. VMesh: Distributed segment storage for peer-to-peer interactive video streaming. *IEEE J. Select. Areas Commun. Spec. Iss.* 25, 9 (Dec. 2007), 1717–31.

Received January 2016; revised April 2016; accepted April 2016