Scaling Social Media Applications into Geo-Distributed Clouds

Yu Wu*, Chuan Wu*, Bo Li†, Linquan Zhang*, Zongpeng Li‡, Francis C.M. Lau*
*Department of Computer Science, The University of Hong Kong, Email: {ywu,cwu,lqzhang,fcmlau}@cs.hku.hk
†Dept. of Computer Science and Engineering, Hong Kong University of Science and Technology, Email: bli@cse.ust.hk ‡Department of Computer Science, University of Calgary, Canada, Email: zongpeng@ucalgary.ca

***Abstract*—Federation of geo-distributed cloud services is a trend in cloud computing which, by spanning multiple data centers at different geographical locations, can provide a cloud platform with much larger capacities. Such a geo-distributed cloud is ideal for supporting large-scale social media streaming applications (*e.g.*, YouTube-like sites) with dynamic contents and demands, owing to its abundant on-demand storage/bandwidth capacities and geographical proximity to different groups of users. Although promising, its realization presents challenges on how to efficiently store and migrate contents among different cloud sites (*i.e.* data centers), and to distribute user requests to the appropriate sites for timely responses at modest costs. These challenges escalate when we consider the persistently increasing contents and volatile user behaviors in a social media application. By exploiting social influences among users, this paper proposes efficient proactive algorithms for dynamic, optimal scaling of a social media application in a geo-distributed cloud. Our key contribution is an online content migration and request distribution algorithm with the following features: (1) future demand prediction by novelly characterizing social influences among the users in a simple but effective epidemic model; (2) one-shot optimal content migration and request distribution based on efficient optimization algorithms to address the predicted demand, and (3) a $\Delta(t)$-step look-ahead mechanism to adjust the one-shot optimization results towards the offline optimum. We verify the effectiveness of our online algorithm by solid theoretical analysis, as well as thorough comparisons with ready algorithms including the ideal offline optimum, using large-scale experiments with dynamic realistic settings on Amazon Elastic Compute Cloud (EC2).**

I. Introduction

The cloud computing paradigm of late enables rapid on-demand provisioning of server resources to applications with minimal management efforts. Most existing cloud systems, *e.g.*, Amazon EC2 and S3, Microsoft Azure, Google App Engine, organize their shared pool of servers from one or a few data centers, and serve their users using different virtualization technologies. The services provided by one individual cloud provider are typically deployed to one or a few geographic regions, prohibiting it from serving application demands equally well from all over the globe. To truly fulfill the promise of cloud computing, a rising trend is to federate disparate cloud services (in separate data centers) from different providers, *i.e.*, interconnecting them based on common standards and policies to provide a universal environment for cloud computing [1], [2]. The aggregate capabilities of a federated cloud would appear to be limitless and can serve a wide range of demands over a much larger geographic span [2].

A geo-distributed federated cloud is ideal for supporting large-scale social media streaming applications. Social network applications (*e.g.*, Facebook, Twitter, Foursquare) are dominating the Internet today, and they are uniting with conventional applications, such as multimedia streaming, to produce new *social media applications*, *e.g.*, YouTube-like sites. Compared with traditional Internet video services, social media applications feature highly dynamic contents and demands, and typically more stringent requirements on response latency in serving viewing requests—since most of their videos are short, *e.g.*, several minutes, a latency of more than a few tens of seconds would be intolerable to a viewer. It is therefore challenging to design and scale a social media application cost-effectively. The conventional approaches use dedicated servers owned by the application providers (*i.e.*, private clouds), or to outsource to a content distribution network (CDN). Geo-distributed clouds provide a much more economic solution: "infinite" on-demand cloud resources meet well with the ever-increasing demand for storage and bandwidth, while capable of absorbing frequent surges of viewing demands on the fly; cloud sites situated in different geographic locations offer efficient services to groups of users in their proximity; elastic charging models of the clouds can significantly cut down operational costs of the application providers.

To realize the potentials of geo-distributed federated clouds, in supporting social media applications, challenges remain to be resolved: How should the social media contents be stored and migrated across different cloud sites, and viewing requests be distributed, such that the operational costs are minimized while the average response delays are bounded according to a pre-set QoS target by the application provider? It may not be too hard to design optimal strategies for the case where the number of contents and the scale of user requests are fixed, which is what a CDN or a cache network is most capable in handling. What is really challenging is to design an *online* algorithm that can make use of cloud resources to accommodate dynamic contents/demands on the fly, and further pursue the optimality achieved by an optimal offline solution with complete knowledge of the system over a long

The research was supported in part by a grant from Hong Kong RGC under the contract HKU 717812E, by a grant from RGC under the contract 615613, by a grant from NSFC/RGC under the contract N_HKUST610/11, by a grant from NSFC under the contract U1301253, and by a grant from ChinaCache Int. Corp. under the contract CCNT12EG01.

time.

Our work proposes such an online algorithm for dynamic, optimal scaling of a social media application in a geo-distributed cloud. Our contributions are as follows:

First, we enable proactive content migration, by predicting future demand based on social influence among the users and correlation across videos. More specifically, a simple but effective epidemic model is built to capture propagation of video views along both social connections (*i.e.*, people view the videos posted or retweeted by their friends) and interest correlations (*e.g.*, people watched a French Open clip may view another one from the Wimbledon).

Second, to serve the predicted demands, we decide on the one-shot optimal content migration and request distribution strategy by formulating the problem as a mixed integer program. We show that efficient solutions to the problem exist, using dual decomposition and linear programming techniques.

Third, a $\Delta(t)$-step look-ahead mechanism is proposed to adjust the one-shot optimization results towards the offline optimality, which gives rise to the online algorithm. We prove the effectiveness of the algorithm using solid theoretical analysis, and demonstrate how the algorithm can be practically implemented in a real-world geo-distributed cloud with low costs. We also design an efficient optimal offline algorithm that derives the offline optimum of the long-term optimization problem, as a benchmark to evaluate performance of our online algorithm.

Finally, performance of our algorithm is evaluated via large-scale experiments under dynamic realistic settings on Amazon EC2. We extensively compare the performance of our online algorithm with that of ready, heuristic dynamic algorithms, as well as against the offline optimum derived by the optimal offline algorithm. The results show that our online algorithm enables high-performance social media applications on a geo-distributed cloud with an operational cost much lower than those achieved by the dynamic heuristics, and close to the offline minimum.

The remainder of this paper is organized as follows. We discuss related work in Sec. II, and present the system model and the offline optimal content migration and request distribution problem in Sec. III. We predict viewing demands and solve the one-shot optimization in Sec. IV. The design of the online algorithm with $\Delta(t)$-step look-ahead and the optimal offline algorithm is given in Sec. V, for which we discuss the evaluation results in Sec. VI. Sec. VII concludes the paper.

II. Related Work

Federation of geo-distributed cloud services is a recent development of cloud computing technologies. Several standardization projects [2] [3] [4] [5] have emerged, which aim to realize a global, interoperable federated cloud ecosystem. For instance, the open data center alliance [2] aims to provide solutions to unify cloud resources from different providers to produce a global-scale cloud platform. The current literature and industry efforts focus on designing inter-connecting standards [1] [6] [7], while our study here, as a complement to the existing work, explores utilization of a geo-distributed cloud platform for efficient application support.

There were a few proposals on migrating applications from conventional private server clusters to the new public cloud platforms. Hajjat *et al.* [8], Sharma *et al.* [9], and Zhang *et al.* [10] advocate migrating enterprise IT applications to exploit the computation and storage capacities of a cloud. Wu *et al.* [11] and Li *et al.* [12] discuss migration of VoD services onto a cloud platform, by exploring demands and user patterns in a conventional VoD application. Pujol *et al.* [13] and Xu *et al.* [14] investigate migration of social network applications, focusing on user profile replication on cloud servers according to their social connections. Different from all these work, our study is the first to explore dynamic migration of the novel social media applications, and to use social *influence* among users for viewing demand prediction; and we target at a solution with over-time optimality guarantee.

Prediction of application behaviour is important for fully exploiting agile resource provisioning of a cloud [15]. The measurement study by Zhou *et al.* [16] reveals the important of related video recommendation on YouTube video viewing counts. Wang *et al.* [17] and Lai *et al.* [18] unveil the correlation between video popularity and the propagation behaviour of links to the videos in a social network via web crawling methodologies. In contrast, our work aims to design a tractable epidemic model for future video demand prediction by fully exploiting the social influences among users and correlations among video contents. Scellato *et al.* [19] exploit geographic information extracted from social cascades to improve multimedia file caching in different CDN sites. A location-aware cache replacement policy is proposed, which ensures that content relevant to a social cascade is kept close to the users who may be interested in it. No content migration across different sites nor request dispatching are investigated. We are going to compare our algorithm with this caching strategy with experiments in Sec. VI.

A substantial body of literature has been devoted to content replication and scheduling in a CDN or cache network [20][21], which mostly targets at relative static scenarios where the contents and user scales are fixed. Our work differs from those work in that we focus on a geo-distributed cloud platform, with significantly different charging models and elastic "pay-per-use" usage patterns, which calls for a more flexible online algorithm. A recent study by Chen *et al.* [22], which appeared in the same venue as the conference version of our work [23], advocates to build CDNs on top of the cloud infrastructure by proposing a set of online and offline heuristics for site replication and distribution path selection. In contrast, our work focuses on content replication and request dispatching in a social media application and our proposed offline and online algorithms exploit the unique social influences in such an application.

In the online algorithm literature [24], paging problems resemble ours from some perspectives, *e.g.*, contents can be migrated among nodes and the access costs depend largely on the distances between the requester and the replica. There have been a variety of work [25] [26] proposing online algorithms, both deterministic and random ones, for the classical paging problems. However, the optimization problem in our work follows closely the realistic parameters of a cloud system and

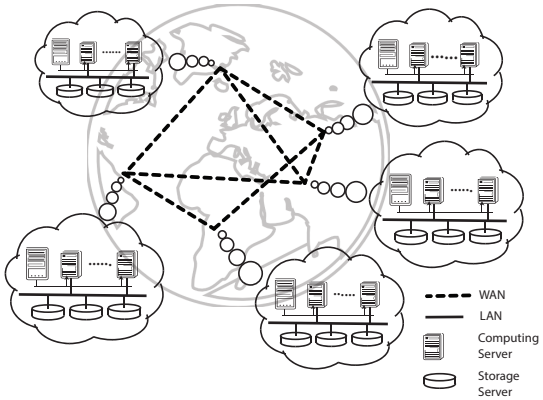

Fig. 1. The geo-distributed cloud model.

is hence much more complicated than the classical paging problems, preventing the application of any ready online algorithms.

III. System Model

A. The Geo-distributed Cloud

We consider a geo-distributed cloud infrastructure (Fig. 1) which consists of multiple disparate *cloud sites* distributed in different geographical locations, and owned by one or multiple cloud service providers. Each cloud site resides in one data center, and contains a collection of interconnected and virtualized servers. A representative structure of servers inside each data center is as follows [27]: There are two categories of servers, *storage servers* to store data files and *computing servers* to support the running and provisioning of *virtual machines* (VMs); all computing and storage servers inside a cloud site are inter-connected with high speed switches and LAN buses. Different cloud sites are connected over a WAN. We investigate the IaaS (Infrastructure as a Service) mode of cloud computing in this work [28].

We assume the computing (storage) servers inside a cloud site have similar hardware configurations, and charge the same prices for usage. Hardware configurations and usage charges are likely to be different across different cloud sites. We take into account the following three types of charges to a cloud consumer: storage cost to keep data on the storage servers, rental fee of VMs to run the application, and charges for incoming/outgoing traffic to/from each cloud site. The former two are charged by usage time on a per unit time rate, and the last one is by traffic volume on a per byte rate. These follow the representative charging models of leading commercial cloud products, such as Amazon EC2 [29] and S3[30].

B. The Social Media Application

In a social media streaming application, registered users generate and upload videos to the servers, and download and view videos uploaded by others. The videos are assumed to be short clips of a few tens or hundreds of mega bytes. Users of the application are interconnected in a social network: besides video browsing and watching, a platform is provided where each user can add other users as friends, post microblogs to comment on videos, and follow microblogs of their friends to watch a video. On the other hand, the system can recommend videos to users (*e.g.*, by listing recommended videos alongside the video currently played) based on such parameters as user location, video types, metadata (tags), top hits, etc. A concrete example of social media application is YouTube enhanced by social networking functions, *i.e.*, a combination of YouTube and twitter (which is an emerging move for YouTube-like applications [16]).

C. The Offline Optimal Content Migration and Request Distribution Problem

The conventional approach to provisioning for this social media application is to use a private server cluster (the application provider's private cloud). We advocate migrating the application into the geo-distributed cloud infrastructure, for better scalability, lower management overhead, and proximity to users. The private cloud may or may not be part of the federated cloud. As a cloud consumer, the application provider deploys its web service on the VMs on the computing servers, and video files in the storage servers.

Our objective is to design an online algorithm to optimally replicate videos onto cloud sites with different charges and proximities to users, and dispatch video requests to the sites such that timely responses at the lowest cost is achieved. We first formulate an *offline optimization* problem which gives the "ideal" optimal strategies for content replication and request dispatching, assuming complete information of the system over the entire time span is known.

Suppose that time is slotted into equal intervals, where $t = 0$ indicates the initial state. Let $C(t)$ denote the set of videos in the social media application at time slot t. We assume that all videos in the system have the same unit size, and the length of a time slot is sufficient for downloading one video at the video playback rate. Let F denote the set of regions that the cloud infrastructure spans, *i.e.*, one region hosts one cloud site. $D_f^{(c)}(t)$ represents the set of users in region f ($f \in F$)[1], who choose to view video c ($c \in C(t)$) in time slot t.

Let $\vec{y}$ and $\vec{\alpha}$ be the optimal decision variables: Binary variable $y_f^{(c)}(t)$ indicates whether a copy of video c should be stored on the cloud site in region f (referred to as cloud site f hereinafter) in time slot t; $\alpha_{jf}^{(c)}(t) \in [0, 1]$ is the portion of $|D_j^{(c)}(t)|$ (the total number of requests for content c from region j at t), to be dispatched to and served by cloud site f.

On cloud site f, p_f is the storage cost per unit size per time slot, m_f is the rental cost of one VM per time slot, and b_f is the outgoing bandwidth cost per unit size. We model the cost incurred for using the cloud platform as follows: (1) The storage cost in time slot t for video c on cloud site f is $y_f^{(c)}(t) \times p_f$. (2) Suppose the number of requests a VM on cloud site f can serve per time slot is n_f. The cost for cloud site f to serve requests from region j for video c in t includes (i) VM rental cost $\frac{\alpha_{jf}^{(c)}(t) \times |D_j^{(c)}(t)|}{n_f} \times m_f$ and (ii) upload bandwidth cost $\alpha_{jf}^{(c)}(t) \times |D_j^{(c)}(t)| \times b_f$. Let $v_f =$

[1]Users residing in regions without deployed cloud sites are considered in sets $D_f^{(c)}(t)$ of regions $f \in F$ that they are geographically closest to.

$\frac{m_f}{n_f} + b_f$ denote the unit cost to serve each request on cloud site f. The cost above can be simplified to $\alpha_{jf}^{(c)}(t) \times |D_j^{(c)}(t)| \times v_f$. (3) Let φ_f denote the migration cost to move one video into cloud site f,[2] which includes bandwidth cost and other management overheads; therefore, $[y_f^{(c)}(t) - y_f^{(c)}(t-1)]^+ \times \varphi_f$ is the potential migration cost for moving video c into cloud site f at t, where $[y_f^{(c)}(t) - y_f^{(c)}(t-1)]^+ = \max\{y_f^{(c)}(t) - y_f^{(c)}(t-1), 0\}$.

The offline optimization to minimize the overall operational cost of the social media application on the geo-distributed cloud over a possibly long time interval, *i.e.*, $[1, T]$, is formulated as follows:

$$\begin{aligned} \min \quad & \mathbb{H}(\vec{y}, \vec{\alpha}) = \textstyle\sum_{t=1}^{T} (\sum_{c \in C(t)} \sum_{f \in F} y_f^{(c)}(t) \times p_f \\ & + \textstyle\sum_{c \in C(t)} \sum_{j \in F} \sum_{f \in F} \alpha_{jf}^{(c)}(t) \times |D_j^{(c)}(t)| \times v_f \\ & + \textstyle\sum_{c \in C(t)} \sum_{f \in F} [y_f^{(c)}(t) - y_f^{(c)}(t-1)]^+ \times \varphi_f) \end{aligned} \tag{1}$$

subject to: (repeat each constraint for $t = 1, \ldots, T$)

$$\begin{aligned} & (a) \quad y_f^{(c)}(t) \in \{0, 1\}, \quad \forall c \in C(t), \forall f \in F, \\ & (b) \quad \alpha_{jf}^{(c)}(t) \le y_f^{(c)}(t), \quad \forall c \in C(t), \forall j \in F, \forall f \in F, \\ & (c) \quad \frac{\sum_{j \in F} \sum_{f \in F} \alpha_{jf}^{(c)}(t) \times |D_j^{(c)}(t)| \times r_{jf}}{\sum_{j \in F} |D_j^{(c)}(t)|} \le \mathcal{R}, \quad \forall c \in C(t), \\ & (d) \quad \sum_{c \in C(t)} \sum_{j \in F} \alpha_{jf}^{(c)}(t) \times |D_j^{(c)}(t)| \le \mathcal{U}_f, \quad \forall f \in F, \\ & (e) \quad \sum_{f \in F} \alpha_{jf}^{(c)}(t) = 1, \quad \forall j \in F, \forall c \in C(t), \\ & (f) \quad 0 \le \alpha_{jf}^{(c)}(t) \le 1, \quad \forall j \in F, \forall f \in F, \forall c \in C(t). \end{aligned}$$

Constraint (a) indicates that video c can be either stored at region f at t or not. Constraints (b), (e) and (f) guarantee that requests would only be dispatched to a cloud site that stores the required video. In constraint (c), r_{jf} represents the round-trip delay between region j and region f (reflecting proximity in between)[3], $\mathcal{R}$ is the upper bound of average response delay per request, set by the application provider; this constraint ensures that the average response delay meets the QoS target. (d) is the bandwidth constraint at each cloud site, where $\mathcal{U}_f$ denotes the maximum reserved bandwidth for this application at cloud site f, in terms of the number of requests to serve. We will address the bandwidth reserving problem as an orthogonal topic in our future work.

In our model, storage and VM capacity limits are not considered at each cloud site, as it is reasonable to assume that these capacities can be provisioned on demand to the application.

To derive optimal solution to the offline optimization (1), complete knowledge about the system over the entire time span is needed, which is apparently not feasible in a dynamic system. We seek to design an online algorithm that pursues this optimal solution (referred to as *optimal offline solution* or *offline optimum* hereinafter) on the fly, with only limited predicted information into the future. In particular, optimization (1) can be decomposed into possibly many *one-shot* optimization problems, each to minimize the operational cost occurred in one time slot. Our idea is to solve the one-shot optimization problem in each time slot, and adjust the derived solutions towards the offline optimum using predicted demands in $\Delta(t)$ time slots in the future.

In what follows, we discuss efficient solutions to the one-shot optimization in Sec. IV, and propose strategies to adjust the one-shot optimum in Sec. V. Important notations in the paper are summarized in Table I for ease of reference.

[2]We assume that there is permanent storage owned by the social media application provider to store one authentic copy of each video, and video replica will be copied from this storage to different cloud sites.

[3]The round-trip delay between each pair of regions can vary from one time slot to the next; we omit (t) from the more rigorous notation $r_{jf}(t)$ for simplification of notation in the paper.

TABLE I
NOTATION

Symbol	Definition
F	Set of regions the geo-distributed cloud spans
$C(t)$	Set of videos in the social media application at t
$y_f^{(c)}(t)$	binary variable: to store video c at cloud site f at t (1) or not (0)
$\alpha_{jf}^{(c)}(t)$	Portion of the total number of requests for video c from region j at t, to be dispatched to cloud site f
$D_f^{(c)}(t)$	Set of users in region f requesting video c at t
p_f	Storage cost per unit size per time slot on cloud site f
v_f	Cost to serve each request on cloud site f (VM rental+bandwidth)
r_{jf}	Round-trip delay between region j and region f
φ_f	Migration cost to move one video into cloud site f
$\mathcal{R}$	Maximum average response delay per request
$\mathcal{U}_f$	Maximum reserved bandwidth at cloud site f
$t_0^{(c)}$	Uploading time of video c
$o^{(c)}$	Uploader of video c
$s^{(c)}(t)$	Number of potential viewers of video c at t
$A^{(c)}(t)$	Set of users who have not watched video c by the end of t
$L^{(c)}(t)$	Set of users that video c is recommended to at t
$E^{(c)}(t)$	Set of users who comment on video c at t
Ω	Set of all registered users
$N(i)$	User i's set of friends
η_c	Initial popularity of video c
γ_c	Decreasing speed of video c' popularity

IV. One-Shot Optimization

The one-shot optimization problem from the offline optimization (1) is as follows, for time slot t:

$$\begin{aligned} \min \quad & \mathbb{F}(\vec{y}(t), \vec{\alpha}(t)) = \textstyle\sum_{c \in C(t)} \sum_{f \in F} y_f^{(c)}(t) p_f \\ & + \textstyle\sum_{c \in C(t)} \sum_{j \in F} \sum_{f \in F} \alpha_{jf}^{(c)}(t) |D_j^{(c)}(t)| v_f \\ & + \textstyle\sum_{c \in C(t)} \sum_{f \in F} [y_f^{(c)}(t) - y_f^{(c)}(t-1)]^+ \varphi_f \\ & \text{subject to: Constraints (a) — (f) in (1),} \end{aligned} \tag{2}$$

where $\vec{y}(t) = (y_f^{(c)}(t), \forall c \in C(t), \forall f \in F)$ and $\vec{\alpha}(t) = (\alpha_{jf}^{(c)}(t), \forall c \in C(t), \forall f \in F, \forall j \in F)$. In time slot $t-1$, we predict the number of upcoming requests for different videos from different regions, *i.e.*, $|D_j^{(c)}(t)|$, for the next time slot t, and solve the above one-shot optimization to derive the best content migration and request distribution strategies for t. This proactive approach is adopted in order to deploy videos in a timely fashion to serve the upcoming requests. We next discuss efficient methods to predict the demand and to solve the one-shot optimization, respectively.

A. Predicting the Number of Viewing Requests

Based on our social media application model in Sec. III-B, potential viewers of video c at t mainly come from two sources: (i) the friends of a user who has watched and commented on the video in her microblog before t, and (ii) the users to whom the system has recommended the video before

t, when they are watching other videos. We predict the number of viewing requests for a video by modeling the propagation of video viewing among users using a model similar to the SIR epidemic model [31].

Let $t_0^{(c)}$ denote the time video c is uploaded by user $o^{(c)}$. $s^{(c)}(t)$ is the number of all potential viewers of video c at time t. $d^{(c)}(t) = \sum_{f \in F} |D_f^{(c)}(t)|$ denotes the number of users who request and view video c at t in the entire system, and $D^{(c)}(t)$ is this set of users. Note that $d^{(c)}(t)$ is different from $s^{(c)}(t)$, in that the latter counts all users who may possibly issue a viewing request (since they belong to category (i) or (ii) above), while the former includes the actually issued ones. Let $A^{(c)}(t)$ be the set of users who have not watched video c by the end of time slot t. Ω represents the set of all registered users in the system, and $N(i)$ is user i's set of friends[4]. $L^{(c)}(t)$ represents the set of users to whom the system recommends video c in t. $E^{(c)}(t)$ is the set of users who comment on video c on her microblog in t.

Measurements of video sharing sites have shown that popularity of a video is typically the highest when it is a new upload, and decreases over time [12] [32]. We employ an exponential decreasing model to describe this phenomenon: we use $\eta_c \times \gamma_c^{(t-t_0^{(c)})}$ to represent the probability that a potential viewer of video c may actually watch the video at t, where factor $\eta_c \in [0,1]$ and $\gamma_c \in [0,1]$ correspond to the initial value and the decreasing speed of video c's popularity, respectively. In practice, both η_c and γ_c can be summarized from historical traces on viewing requests for video c, and are dynamically calibrated with the propagation of that video.

Without loss of generality, we assume that a user will not issue viewing requests again for a video that she has requested before, and the first batch of viewing requests come at $t_0^{(c)}+1$, but not in $t_0^{(c)}$ when the video is newly shared. The epidemic model to describe the propagation of video viewing in the system is as follows, where $t > t_0^{(c)}$:

$$\begin{aligned}
&\text{(i)}\ \ s^{(c)}(t_0^{(c)}) = 0,\\
&\text{(ii)}\ \ d^{(c)}(t_0^{(c)}) = 0,\\
&\text{(iii)}\ \ A^{(c)}(t_0^{(c)}) = \Omega \backslash \{o^{(c)}\},\\
&\text{(iv)}\ \ s^{(c)}(t) = s^{(c)}(t-1) - d^{(c)}(t-1)\\
&\qquad + |\cup_{i \in E^{(c)}(t-1)} (N(i) \cap A^{(c)}(t-1)) \cup L^{(c)}(t-1)|,\\
&\text{(v)}\ \ d^{(c)}(t) = s^{(c)}(t) \times \eta_c \times \gamma_c^{(t-t_0^{(c)})},\\
&\text{(vi)}\ \ A^{(c)}(t) = A^{(c)}(t-1) \backslash D^{(c)}(t).
\end{aligned} \tag{3}$$

The rationale is as follows: When video c is uploaded at $t_0^{(c)}$, no other users than $o^{(c)}$ have watched it (Eqn. (i)—(iii) in (3)). The potential set of viewers at t is derived in Eqn. (iv) by excluding those who have viewed video c at $t-1$ from the previous set of potential viewers ($s^{(c)}(t-1) - d^{(c)}(t-1)$), and adding the newly emerged potential viewers, *i.e.*, the friends of those commented on c at $t-1$, who have not yet viewed it ($\cup_{i \in E^{(c)}(t-1)} N(i) \cap A^{(c)}(t-1)$), and users that the system recommends c to at $t-1$ ($L^{(c)}(t-1)$). Since a potential viewer may not actually watch the video, in Eqn. (v) the number of actual viewers is estimated by multiplying the number of potential viewers by probability $\eta_c \times \gamma_c^{(t-t_0^{(c)})}$. Finally, the set of users who have never watched the video by the end of t will be reduced by the set who have viewed it at t, described by Eqn. (vi).

Predict all viewing requests: We predict the total number of actual viewers for video c in the system, *i.e.*, $d^{(c)}(t)$, based on Eqn. (iv) and (v), using known information at $t-1$: the number of potential viewers ($s^{(c)}(t-1)$), the number of actual viewing requests ($d^{(c)}(t-1)$), the users who comment on video c ($E^{(c)}(t-1)$) and their neighbors who have not viewed the video, as well as the users receiving system recommendation ($L^{(c)}(t-1)$).

Map to geographic regions: Next, we calculate the number of potential viewers in region f, $s_f^{(c)}(t)$, using an equation similar to Eqn. (iv), which only counts users in f in each term:
$s_f^{(c)}(t) = s_f^{(c)}(t-1) - |D_f^{(c)}(t-1)| + |\cup_{i \in E^{(c)}(t-1)} (N_f(i) \cap A^{(c)}(t-1)) \cup L_f^{(c)}(t-1)|$, where $N_f(i)$ and $L_f^{(c)}(t-1)$ represent i's neighbors in region f and users receiving system recommendation in region f, respectively. We can then estimate the number of actual viewing requests for video c from region f as $d_f^{(c)}(t) = \frac{s_f^{(c)}(t)}{s^{(c)}(t)} \times d^{(c)}(t)$.

B. Solving the One-Shot Optimization

Define

$$p_f(t) = \begin{cases} p_f, & if \quad y_f^{(c)}(t-1) = 1, \\ p_f + \varphi_f, & if \quad y_f^{(c)}(t-1) = 0, \end{cases} \quad \forall f \in F.$$

When $y_f^{(c)}(t-1)$ (video replication in $t-1$) is given, $p_f(t)$ is a constant. We can rewrite one-shot optimization (2) as follows:

$$\begin{aligned}
&\min \quad \mathbb{F}(\vec{y}(t), \vec{\alpha}(t)) = \textstyle\sum_{c \in C(t)} \sum_{f \in F} y_f^{(c)}(t) p_f(t)\\
&+ \textstyle\sum_{c \in C(t)} \sum_{j \in F} \sum_{f \in F} \alpha_{jf}^{(c)}(t) |D_j^{(c)}(t)| v_f\\
&s.t. \begin{cases} \vec{y}(t) \in \mathbb{C}_1, \\ \vec{\alpha}(t) \in \mathbb{C}_2, \\ \alpha_{jf}^{(c)}(t) - y_f^{(c)}(t) \le 0, \forall c \in C(t), \forall j \in F, \forall f \in F, \end{cases}
\end{aligned} \tag{4}$$

where $\mathbb{C}_1$ is the set defined by constraint (a) in (1), and $\mathbb{C}_2$ is the set defined by linear constraints (c)—(e) in (1). This optimization problem is a mixed integer program. Nevertheless, we next show that an efficient solution indeed exists through *dual decomposition* [33].

We derive the dual problem of (4) by relaxing its last constraint group. Associating dual variables $\vec{\lambda}(t) = (\lambda_{jf}^{(c)}(t))$ with those constraints, the Lagrangian is:

$$\begin{aligned}
&\mathbb{L}(\vec{y}(t), \vec{\alpha}(t), \vec{\lambda}(t))\\
&= \textstyle\sum_{c \in C(t)} \sum_{f \in F} y_f^{(c)}(t) (p_f(t) - \sum_{j \in F} \lambda_{jf}^{(c)})\\
&\quad + \textstyle\sum_{c \in C(t)} \sum_{j \in F} \sum_{f \in F} \alpha_{jf}^{(c)}(t) (|D_j^{(c)}(t)| v_f + \lambda_{jf}^{(c)}).
\end{aligned} \tag{5}$$

The dual function is then as follows, which is separable:

$$g(\vec{\lambda}(t)) = g_1(\vec{\lambda}(t)) + g_2(\vec{\lambda}(t))$$

where

$$\begin{aligned}
&g_1(\vec{\lambda}(t)) = \min \textstyle\sum_{c \in C(t)} \sum_{f \in F} y_f^{(c)}(t) (p_f(t) - \sum_{j \in F} \lambda_{jf}^{(c)}) \quad (A)\\
&s.t. \quad \vec{y}(t) \in \mathbb{C}_1,\\
&g_2(\vec{\lambda}(t)) = \min \textstyle\sum_{c \in C(t)} \sum_{j \in F} \sum_{f \in F} \alpha_{jf}^{(c)}(t) (|D_j^{(c)}(t)| v_f + \lambda_{jf}^{(c)})\\
&s.t. \quad \vec{\alpha}(t) \in \mathbb{C}_2. \quad (B)
\end{aligned}$$

[4]We only consider fixed friendship graph and ignore newly registered users.

TABLE II
ALGORITHM SKETCH TO SOLVE ONE-SHOT OPTIMIZATION IN (2)

Repeat
Solve subproblems (A) and (B) (in parallel)
Find optimal content replication $\vec{y}(t)$ that solves $g_1(\vec{\lambda}(t))$
Find optimal request distribution $\vec{\alpha}(t)$ that solves $g_2(\vec{\lambda}(t))$
Update dual variables by $\lambda_{jf}^{(c)}(t) := \lambda_{jf}^{(c)}(t)+\beta_k(\alpha_{jf}^{(c)}(t)-y_f^{(c)}(t))$, $\forall c \in C(t), \forall j \in F, \forall f \in F$

The dual problem is: $\max g(\vec{\lambda}(t))$ s.t. $\vec{\lambda}(t) \succeq 0$.

The dual problem can be solved by the subgradient algorithm [33], which gives the optimal primal variable values as well (*i.e.*, the optimal solution to one-shot optimization (2)). The sketch of the subgradient algorithm is given in Table. II, which has a nice intuitive interpretation as follows:

We start with any initial non-negative dual variable values $\lambda_{jf}^{(c)}(0)$. In the k^{th} iteration, given current values of $\lambda_{jf}^{(c)}(t)$'s, we solve the optimal content replication subproblem (A) and the optimal request dispatching subproblem (B) independently, and derive the content replication and request dispatching strategies, *i.e.*, $y_f^{(c)}(t)$'s and $\alpha_{jf}^{(c)}(t)$'s, respectively. Subproblem (B) is a linear program and can be solved efficiently using polynomial-time algorithms [34]. Integer program (A) can be solved efficiently too: we relax the integer constraints $y_f^{(c)}(t) \in \{0,1\}$ in $\mathbb{C}_1$ to $0 \le y_f^{(c)}(t) \le 1$ ($\forall c \in C(t), \forall f \in F$), and prove that the optimal solution to the integer program can be instantly derived from the optimal solution to the resulting linear program in Lemma 1.

Lemma 1. *There exists an integer optimal solution to the relaxed linear program of the integer subproblem (A), which is the optimal solution to the integer subproblem (A).*

The proof of the lemma is given in Appendix A.

In Table II, after efficiently solving the two subproblems, we update the value of dual variables. Here, $\beta_k = \frac{1}{k}$, which is a step size used in the k^{th} iteration. $\lambda_{jf}^{(c)}$ can be seen as the price of violating constraint $\alpha_{jf}^{(c)}(t) - y_f^{(c)}(t) \le 0$. If it is violated, *i.e.*, the solution to subproblem A indicates that requests for video c are to be dispatched to region f ($\alpha_{jf}^{(c)}(t) > 0$) while the solution to subproblem B states that video c is not to be stored in region f ($y_f^{(c)}(t) = 0$), then $\lambda_{jf}^{(c)}$ is increased, such that content replication and request dispatching will be adjusted in the next iteration towards satisfaction of this constraint.

The steps repeat until converging to the optimal decisions which satisfy all constraints and minimize the aggregate operational cost in time slot t in (2). We have therefore derived an efficient algorithm to solve the one-shot optimization.

V. ONLINE ALGORITHM WITH $\Delta(t)$-STEP LOOK-AHEAD

Although one-shot optimal decisions can be efficiently made in any single time slot, they do not guarantee the optimality of the offline optimization (1) over a possibly long time. Let $\vec{y}^* = (y_f^{*(c)}(t), \forall c \in C(t), \forall f \in F, t = 1, \ldots, \infty)$ and $\vec{\alpha}^* = (\alpha_{jf}^{*(c)}(t), \forall c \in C(t), \forall f, j \in F, t = 1, \ldots, \infty)$ denote the *optimal offline solution* for (1). For example, suppose video c is stored in region f at $t-1$, and removing c from f is cost-optimal at t ($y_f^{(c)}(t) = 0$) according to the one-shot optimization (*e.g.*, because the demand for c in f drops significantly at t); however, it is possible that c should remain in f at t and for a number of following time slots in the offline optimum ($y_f^{*(c)}(t) = 1$), since the demand for the video in the region will rise again soon, and keeping video c there could have saved the migration cost.

We first design an optimal offline algorithm to derive the offline optimum based on the one-shot optimization problems, with complete knowledge of the system in the entire span. We next explore dependencies among video replication decisions across consecutive time slots, and design a practical online algorithm to improve solutions towards offline optimum.

A. An Optimal Offline Algorithm

The algorithm is designed using dynamic programming. Let $\mathcal{P}(t)$ denote the set of all possible content replication strategies at time slot t:

$$\mathcal{P}(t) = \{\vec{y}(t) \mid y_f^{(c)}(t) \in \{0,1\}, \forall c \in C(t), f \in F\}.$$

Let $Opt(t, \vec{y}(t))$ denote the optimal cost from the first time slot to t with $\vec{y}(t)$ as the content replication decision at t. The algorithm begins with $Opt(1, \vec{y}(1)) = \min_{\vec{\alpha}(1):\text{(b)-(f) in (1)}} \mathbb{F}(\vec{y}(1), \vec{\alpha}(1)), \forall \vec{y}(1) \in \mathcal{P}(1)$, and computes optimal costs in later time slots ($t > 1$) inductively:

$$Opt(t, \vec{y}(t)) = \min_{\substack{\vec{y}(t-1) \in \mathcal{P}(t-1) \\ \vec{\alpha}(t) : \text{(b)-(f) in (1)}}} \{Opt(t-1, \vec{y}(t-1)) + \mathbb{F}(\vec{y}(t), \vec{\alpha}(t))\}. \quad (6)$$

Given $\vec{y}(t)$, $Opt(t, \vec{y}(t))$ computes the minimum cumulative cost from time slot 1 up to t, by choosing among all possible content replication decisions $\vec{y}(t-1) \in \mathcal{P}(t-1)$ in $t-1$, and all feasible request dispatching decisions $\vec{\alpha}(t)$ in t. The term $Opt(t-1, \vec{y}(t-1))$ is the minimum cumulative cost in $[1, t-1]$ with the specific $\vec{y}(t-1)$ as the content replication decision at $t-1$; the term $\mathbb{F}(\vec{y}(t), \vec{\alpha}(t))$ is the cost incurred in time slot t. Here $\vec{y}(t-1)$ is related to $\mathbb{F}(\vec{y}(t), \vec{\alpha}(t))$, since it decides the potential migration cost at t. If there is no feasible solution to the minimization problem in (6), then we set $Opt(t, \vec{y}(t)) = +\infty$.

The rationale of the dynamic programming approach is as follows. At each time t, fixing the content replication strategy, we trace back and examine each possible content replication strategy in time $t-1$, by adding the cost incurred in t to the minimum cumulative cost up to $t-1$; that is, we compute the cumulative costs up to t in $|\mathcal{P}(t-1)|$ cases (corresponding to these many content replication strategies in $t-1$), and then decide the minimum cumulative cost up to t via the best content replication strategy in $t-1$. Eventually when the computation up to time slot T is completed, the minimum overall cost of the system in $[1, T]$, *i.e.*, the optimal objective function value of the offline optimization problem in (1), is given by

$$C_{opt} = \min_{\vec{y}(T) \in \mathcal{P}(T)} Opt(T, \vec{y}(T)).$$

The optimal content replication decision in time slot T is $\vec{y}^*(T) = \arg\min_{\vec{y}(T)\in\mathcal{P}(T)} Opt(T, \vec{y}(T))$, and the optimal request dispatching strategy $\vec{\alpha}^*(T)$ is the one leading to $Opt(T, \vec{y}^*(T))$ by solving (6). The optimal content replication and request dispatching decisions in previous time slots can be derived accordingly, by tracing the optimal decision path back.

Theorem 1. *Consider solving the one-shot optimization problem in (2) in each time slot t with given replication decision $\vec{y}^*(t)$, to derive the optimal request dispatching strategy $\vec{\alpha}^*(t)$, as one atomic operation. The optimal offline algorithm to compute the offline optimum of (1) has a computation complexity of $O(T2^{2|F|\cdot\max_{t\in[1,T]}|C(t)|})$.*

The proof is given in Appendix B. The optimal offline algorithm designed in this section is to serve as a benchmark in performance evaluation. We will compare the offline optimum derived by this algorithm with the cost achieved by our online algorithm, to be discussed next.

B. An Online Algorithm Pursuing Offline Optimality with $\Delta(t)$-step Look-ahead

We next design an efficient online algorithm, which makes decisions in each time slot with only limited predicted information into the future. The basic idea is that, at each time slot $t-1$, we solve the one-shot optimization (2) for the next time slot t, and then adjust the one-shot optimal solution towards the offline optimum. In the following discussions, we focus on content replication strategy ($y_f^{(c)}(t)$'s), knowing that request distribution strategy ($\alpha_{jf}^{(c)}(t)$'s) can be determined accordingly by solving (2), given the content replication strategy. There are two possible replication decisions for video c in region f at t: $y_f^{(c)}(t) = 1$ (caching the video) and $y_f^{(c)}(t) = 0$ (not caching the video), respectively.

(i) If $y_f^{(c)}(t) = 1$ is the derived one-shot optimal decision, we argue that it is also offline optimal to store c in f at t:

Lemma 2. *Given replication decisions at $t-1$,* i.e., *$\vec{y}(t-1)$, if solving one-shot optimization (2) for t gives $y_f^{(c)}(t) = 1$,* i.e., *video c should be stored in region f at t, then in the optimal offline solution, we have $y_f^{*(c)}(t) = 1$.*

The rationale is intuitive: If one-shot optimization gives $y_f^{(c)}(t) = 1$, it shows that caching c in f is desirable to address requests at t, even if storage and possibly migration cost would be incurred. In the offline optimum where future demands are considered, if c is still needed in f in later time slots, storing c there at t is more cost-effective than removing it; even if c is not needed in f later, caching it there is the best strategy for t at least — in both cases, $y_f^{*(c)}(t) = 1$. Rigorous proof of the lemma is given in Appendix C.

(ii) If $y_f^{(c)}(t) = 0$ according to the one-shot optimization, we need to be more cautious, judge whether it is offline optimum by looking ahead for a few time slots, and adjust the decision if we are (almost) sure that it is not. Our adjustment mechanism below focuses on cases that the effect of changing $y_f^{(c)}(t)$ is isolated, *i.e.*, it does not affect video c's deployment in other regions $f'(\neq f)$ in $t+1$ after solving the one-shot optimization for $t+1$, as in these cases we can prove the correctness of our adjustment.

Let $\Delta(t) \geq 0$ denote the number of look-ahead time slots beyond t, whose viewing demands we need to learn in order to decide whether adjusting $y_f^{(c)}(t)$ from 0 to 1 is more cost beneficial over time. We will show how we set $\Delta(t)$ soon. Suppose the number of viewing requests in those $\Delta(t)$ time slots can be predicted[5] or known, *e.g.*, based on summarized daily patterns. According to $y_f^{(c)}(t) = 0$, we calculate the one-shot optimal solutions in $t+1$, $t+2$, ..., by solving (2) for the respective times. Suppose after δt intervals, the one-shot optimum $y_f^{(c)}(t+\delta t)$ becomes 1, *i.e.*, demands arise and video c should be cached in f at $t+\delta t$. If we use $\vec{y}_f^{(c)}[t, \delta t] = (y_f^{(c)}(t), \ldots, y_f^{(c)}(t+\delta t))$ to denote replication decision variables of video c in region f during t to $t+\delta t$, then strategy sequence $\vec{y}_f^{0(c)}[t, \delta t] = (0, 0, \ldots, 0, 1)$ corresponds to one-shot optimal solutions during t to $t+\delta t$ when $y_f^{(c)}(t) = 0$.

If we adjust $y_f^{(c)}(t)$ from 0 to 1 and solve one-shot optimization in the subsequent δt time slots, we can obtain another strategy sequence $\vec{y}_f^{1(c)}[t, \delta t]$. We argue that $y_f^{1(c)}(t+\delta t) = 1$ in this sequence based on the following lemma.

Lemma 3. *Given replication decisions of other videos and video c in other regions, if one-shot optimal solution is to cache c in f in t,* i.e., *$y_f^{(c)}(t) = 1$, by assuming c is not there in $t-1$,* i.e., *$y_f^{(c)}(t-1) = 0$, then $y_f^{(c)}(t) = 1$ is the one-shot optimum no matter whether $y_f^{(c)}(t-1)$ is indeed 0 or 1.*

Proof of Lemma 3 is given in Appendix D. Since $y_f^{0(c)}(t+\delta t) = 1$ is the one-shot optimum at $t+\delta t$ when $y_f^{0(c)}(t+\delta t-1) = 0$, then $y_f^{1(c)}(t+\delta t) = 1$ no matter whether $y_f^{1(c)}(t+\delta t-1)$ is 1 or 0. Therefore, at most δt time slots after adjusting $y_f^{(c)}(t)$ from 0 to 1, the replication strategy sequences $\vec{y}_f^{0(c)}[t, \delta t]$ and $\vec{y}_f^{1(c)}[t, \delta t]$ merge. In fact, the two sequences may merge sooner, *i.e.*, $\delta t' (< \delta t)$ slots after the adjustment, if it turns out $y_f^{1(c)}(t+\delta t') = 0$, and then all subsequent $y_f^{1(c)}(t+\delta t'+1)$, ..., $y_f^{1(c)}(t+\delta t-1)$ will be 0. Hence, when evaluating the impact of $y_f^{(c)}(t)$'s adjustment on cost change, we only need to compare the change of total cost during t to $t+\min(\delta t, \delta t')$, when the two replication strategy sequences diverge, but not afterwards when they merge. The number of look-ahead time slots, $\Delta(t)$, is then set to be $\min(\delta t, \delta t')$.

Let $\mathbb{G}_f^{(c)}(\vec{y}_f^{1(c)}[t, \Delta(t)], \vec{y}_f^{0(c)}[t, \Delta(t)])$ denote the cost difference during t to $t+\Delta(t)$ when adopting the above two replication strategy sequences, respectively. It can be calculated as

$$\begin{aligned}&\mathbb{G}_f^{(c)}(\vec{y}_f^{1(c)}[t, \Delta(t)], \vec{y}_f^{0(c)}[t, \Delta(t)])\\ =&\sum_{\tau=t}^{t+\Delta(t)} \{\mathbb{F}(y_f^{1(c)}(\tau)) - \mathbb{F}(y_f^{0(c)}(\tau))\}.\end{aligned}$$

If $\mathbb{G}_f^{(c)}(\vec{y}_f^{1(c)}[t, \Delta(t)], \vec{y}_f^{0(c)}[t, \Delta(t)]) < 0$, adjusting $y_f^{(c)}(t)$

[5]The prediction can be done following our epidemic model in (3), or using other regression techniques [35].

Algorithm 1 An online algorithm with $\Delta(t)$-step Look-ahead

Input: $\vec{y}(t-1)$, $D(t-1)$, $L(t-1)$, $E(t-1)$.
Output:$\vec{y}(t), \vec{\alpha}(t)$.

Estimate number of viewers $d_j^{(c)}(t)$, $\forall j \in F, \forall c \in C(t)$;
Derive the one-shot optimum $y_f^{(c)}(t)$ and $\alpha_{jf}^{(c)}(t)$, $\forall j, f \in F, c \in C(t)$;
for video $c \in C(t)$ **do**
  Form subset of regions $\Psi = \{f | y_f^{(c)}(t) = 0\}$;
  for region $f \in \Psi$ **do**
    $\Delta(t) = 1$;
    while $\Delta(t) \leq W_{thresh}$ **do**
      Derive one-shot optimum $y_{f'}^{(c)}(t + \Delta(t))$, $\forall f' \neq f$, based on $y_f^{(c)}(t) = 0$ and $y_f^{(c)}(t) = 1$, respectively;
      if $y_{f'}^{(c)}(t+\Delta(t))$ derived in the two cases are different for any $f' \neq f$ **then**
        break;
      end if
      if $y_f^{0(c)}(t + \Delta(t)) = y_f^{1(c)}(t + \Delta(t))$ **then**
        if $\mathbb{G}(\vec{y}^{1(c)}[t, \Delta(t)], \vec{y}^{0(c)}[t, \Delta(t)]) < 0$ **then**
          Set $y_f^{(c)}(t) = 1$;
        end if
        break;
      end if
      $\Delta(t) + +$;
    end while
    Derive $\alpha_{jf}^{(c)}(t), \forall j \in F, f \in F$, based on adjusted $y_f^{(c)}(t)$'s;
  end for
end for

from 0 to 1 reduces the cost in the long run; otherwise, we should retain $y_f^{(c)}(t) = 0$.

We note that $\Delta(t)$ could be quite large or it is possible that $y_f^{0(c)}(t+\delta t) = 1$ never happens when $\delta t \to \infty$. To handle both cases, we set a threshold W_{thresh} to the number of look-ahead steps: if sequences $\vec{y}_f^{0(c)}(t, W_{thresh})$ and $\vec{y}_f^{1(c)}(t, W_{thresh})$ still diverge after W_{thresh} steps, we will just retain $y_f^{(c)}(t) = 0$.

An online algorithm in Algorithm 1 is designed to adjust one-shot optimal solutions towards offline optimum, following the above discussions. Theorem 2 guarantees that Algorithm 1 can derive a solution closer to the offline optimum, than a solution that consists of one-shot optimum in individual time slots.

Theorem 2. *Given the predicted numbers of viewing requests within the next $\Delta(t)$ time slots, Algorithm 1 improves the one-shot optimal solution at each time slot t to one achieving a lower overall operational cost over the system span T.*

Proof of Theorem 2 is given in Appendix. E.

C. Practical Implementation of the Online Algorithm

We briefly discuss how our online Algorithm 1, together with demand prediction and one-shot optimization modules, can be practically implemented in a real-world system. The algorithm can be deployed on the tracker server(s) in the social media application, which is (are) responsible for receiving users' requests and dispatching them to the cloud cites. Key modules of the algorithm are illustrated in Fig. 2.

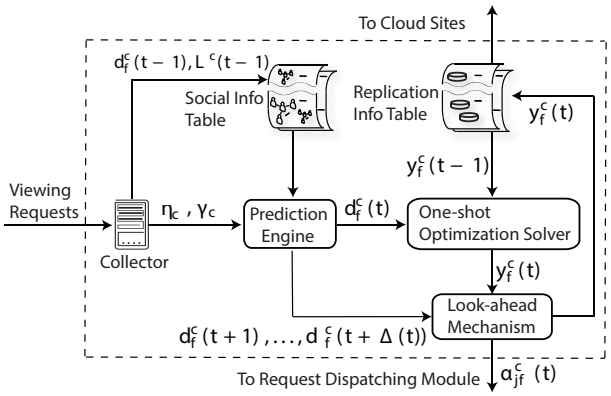

Fig. 2. Key modules in online algorithm implementation.

During each interval t, the *Collector* collects the number of requests for each video from received viewing requests, the friend relationship among users and their geographic distribution, as well as the list of users that the social media system recommends a video to. All these are stored in a *social information table*, as shown in Fig. 2. Based on statistics collected over time, the collector also adjusts the estimates for γ_c and η introduced in Sec. IV-A. The summarized statistics are fed into the *Prediction Engine*, which estimates the number of viewing requests for each video in the upcoming time slot. With the demand prediction from the prediction engine and current video replication status from the *replica information table*, the *One-Shot Optimization Solver* solves the one-shot optimization (2). The *Look-ahead Mechanism* reads in the solution from the one-shot solver and adjusts them towards offline optimality following Algorithm 1. The resulting content replication decisions are sent to the cloud sites, for them to pre-deploy videos and VMs in cases of increased demands and remove videos with decreased demands; request distribution strategies are employed by the social media application to dispatch upcoming requests to different cloud sites.

A number of practical concerns may arise when running the algorithm in real-world social media platforms:

Update frequency. Our algorithm runs periodically. As hourly resource rental is commonly supported in cloud systems [29], the algorithm can be run at intervals of a few hours.

Initial deployment of videos. For a newly uploaded video, a default strategy is to store it in the cloud site closest to the uploader. From this time onwards, the video is included in calculation of the optimal replication strategies.

Large numbers of videos. Social media application may host a large number of videos, which increases over time. Though all videos are included in our optimization formulations, our algorithm is flexible in the set of videos to attend to in each run: A closer investigation of optimization (1) reveals that the replication decisions of one video is largely decoupled from those of other videos. Therefore, we can optimize the replication of a subset of videos in each time slot, but not necessarily all of them. For example, viewing demands of popular videos may expand quickly across regions; we may update their replication at higher frequencies, while dealing with unpopular videos at longer intervals.

Accuracy of multi-step prediction. Our algorithm requires

$\Delta(t)$-step prediction. In fact, as long as the prediction can roughly estimate the evolution trend of viewer populations (*e.g.*, in cases of apparent daily patterns shown by many measurements [36] [37]), our algorithm provides nice guidelines for optimal pre-deployment of videos.

VI. Performance Evaluation

We evaluate the performance of our online algorithm, by building a prototype system on Amazon Elastic Compute Cloud (EC2) [29], under realistic settings.

A. Prototype Implementation and Experimental Settings

We create a geo-distributed cloud by emulating a cloud site using an Amazon "High-CPU Medium Instance" (1.7 GB Ram, 5 EC2 Compute Units) in each of the following 8 regions: Northern Virginia, Oregon, Northern California, Ireland, Singapore, Tokyo, Sydney, and Sao Paulo. The round-trip delays (RTT) between each pair of cloud sites are the real-life measured values of the dispersed instances. Different charges are applied in the 8 cloud sites, as given in Table III.[6] The prices are set based on the charging model of Amazon Web Services [29][30], with minor adjustments.

One extra "Micro Instance" (613 MB Ram, 2 EC2 Compute Units) is provisioned in each region to simulate the group of users located in the region, which produces viewing requests to dispatch to the cloud sites. The RTT between a user and a cloud site is 20 ms (manually injected) if they are in the same region, or the real-world measured values otherwise. The targeted maximal average response delay per request, $\mathcal{R}$, is set to 150 ms, since a latency up to 200ms will deteriorate the user experience significantly [38]. Another "High-Memory Extra Large Instance" (17.1 GB Ram, 6.5 EC2 Compute Units) is created as the tracker server, implementing the *Collector*, *Prediction Engine*, *One-Shot Optimization Solver*, and *Look-ahead Mechanism* discussed in Sec. V.

In our experiments, each time slot is 1 hour long, the same as the provisioning granularity of Amazon EC2 instances. A user relationship matrix $\mathbf{U}$ is specified to define how users are socially connected, *i.e.*, $\mathbf{U_{ij}} = \mathbf{1}$ denotes users i and j are friends, and $\mathbf{U_{ij}} = \mathbf{0}$ otherwise. Another user-content matrix $\mathbf{V}$ keeps track of the users' viewing activities, *i.e.*, $\mathbf{V_{ij}} = \mathbf{1}$ denotes that user i has viewed video j and $\mathbf{V_{ij}} = \mathbf{0}$ otherwise. The number of friends of each user follows a lognormal distribution [39], 80% of which are from the same region where the user resides. To emulate a highly dynamic online social UGC system, for each hour, 3% brand new videos are uploaded to the system by users located in an 'active' region — where the local time is between 9am and 9pm in a day, and the number of viewing requests issued follows the well-known daily patterns [37], where most of the initial viewers of the videos are friends from the same regions of the uploaders. The videos are evenly divided into four types, and each video is 100M-byte long. We generate synthesized traces that describe the evolution of popularity and propagation of each video over time, by following closely patterns revealed in the measurement work [40] and [41], respectively. Besides its propagation following the social relationship among users, each video is also recommended to 0.5% of all users in the system in each hour, who have recently watched a video of the same type. We assume each viewer of a video immediately comments on the video after watching it. Due to the prohibitive traffic cost among EC2 instances, the total number of emulated users in the system is limited to $10,000$ and the initial number of videos is 60. We run the system for over 100 hours.

[6]Large migration costs φ_f are set to capture the large management overheads incurred during content migration.

B. Prediction accuracy

We first investigate effectiveness of our epidemic model for forecasting future viewing demands, by comparison against ARIMA, a widely used model for non-stationary time series prediction [42]. In our epidemic model, we set the values of η_c and γ_c for each video c by matching the resulting evolution of the video popularity with that captured by the traces. We found our model matches the traces best when η_c is set to a value around 0.5 and γ_c is chosen in the range of $[0.9, 0.99999]$, for each video c. When fitting an ARIMA model, we collected 96-hours' user requests in a single dry run. The original series of the number of requests becomes stationary after being differenced twice, and we therefore chose an ARIMA(p,2,q) model; and after carefully checking the partial autocorrelations, an autoregressive model of $p = 3$ and $q = 0$ is applied.

In Fig. 3, the solid blue curve plots the actual viewing request number in a time span of 48 hours, following the synthesized traces we applied. The dotted red curve corresponds to the ARIMA prediction results, using the ARIMA(3,2,0) model. The black square dots represent the prediction results using our epidemic model for 5 consecutive time slots, made at the time slots marked by '+': *e.g.*, the first five square dots are prediction results done at $t = 1$ for the next 5 time slots, the next batch of five square dots are prediction results done at $t = 14$ for the next 5 time slots, and so on. For better readability of the figure, we only show the prediction results made at selected time slots of $t = 1$, 14, 27 and 40, respectively. We can observe that predicted numbers using our epidemic model follow the actual numbers quite well, especially within a 4-hour look-ahead window (*i.e.*, the first four square dots in each batch are well aligned with the blue curve). However, the ARIMA model fails to capture the social influence among users and performs poorly.

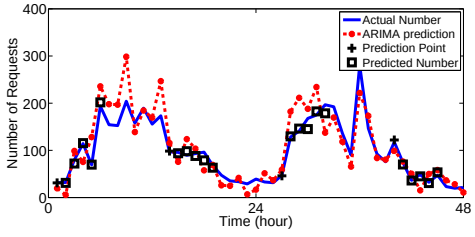

Fig. 3. Evolution of popularity of a sample video.

TABLE III
CONFIGURATIONS OF 8 GEO-DISTRIBUTED CLOUD SITES

	Northern Virginia	Oregon	Northern California	Ireland	Singapore	Tokyo	Sydney	Sao Paulo
φ_f ($ per video)	6.66	7.44	7.20	7.80	7.50	7.11	7.74	6.96
p_f ($ per byte per hour)	0.599	0.559	0.574	0.620	0.562	0.580	0.598	0.576
v_f ($ per request)	0.038	0.035	0.038	0.040	0.039	0.038	0.035	0.034
$\mathcal{U}_f$ (requests per hour)	8,800	7,300	9,100	9,400	8,100	8,000	7,800	87,00

C. Impact of Look-ahead Window Size

We next investigate the performance of our online algorithm when different look-ahead window sizes are employed, *i.e.*, W_{thresh} in Alg. 1. Fig. 4 plots cost savings, *i.e.*, cost incurred with one-shot optimal solutions minus cost with our online algorithm, in each time slot when different maximal window sizes W_{thresh} are used in our look-ahead mechanism. To better illustrate the observations from Fig. 4, Fig. 5 plots the corresponding cost saving percentage when the look-ahead window is adjusted, *e.g.*, $W_{thresh}2 \rightarrow 3$ represents that the look-ahead window size is adjusted from 2 to 3, and the corresponding cost saving percentage is computed as the cost of our online algorithm with $W_{thresh} = 3$ minus the cost of our online algorithm with $W_{thresh} = 2$, and then divided by the later. We observe that a larger window may give larger cost savings, but the gap decreases with the increase of window size, *e.g.*, $W_{thresh} = 2$ or 3 achieve similar costs over time. All these promise that a small look-ahead window is enough to achieve good cost savings in realistic environments. In our following experiments, we will use a look-ahead window size $W_{thresh} = 2$ as the default.

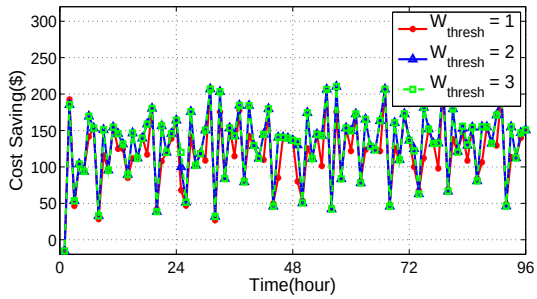

Fig. 4. Evolution of cost saving between our online algorithm and the one-shot optimum: different window sizes.

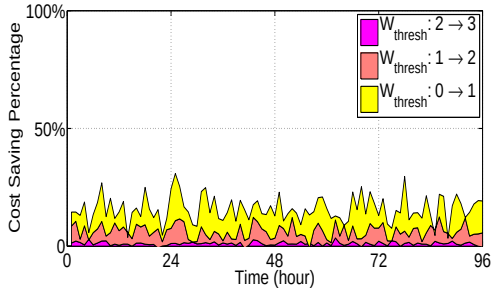

Fig. 5. Evolution of cost saving percentages with different window sizes in our online algorithm.

D. Performance Comparison with Other Algorithms

We compare the performance of our online lookahead algorithm against other potential solutions, including a simple CDN algorithm, a smart CDN algorithm, the one-shot optimum algorithm and the offline optimal algorithm.

▷*Simple CDN*: It replicates a copy of each video in each cloud site at all times. User requests are routed to any cloud site with sufficient bandwidth, as long as the latency constraints are met.

▷*Smart CDN*: This algorithm resembles the one proposed by Scellato *et al.* [19], except that we further consider content migration costs as well as the capacity constraint in each individual cloud site: Upon requests from users in a region, a copy of the requested video will be replicated in an on-demand fashion to the cloud site closest to the social cascade, which has sufficient upload bandwidth.

▷*One-shot Optimum*: The algorithm uses one-shot optimal solutions in each time slot for video replication and request dispatching, such that the cost is minimized in individual time slots.

▷*Offline Optimum*: It carries out the optimal offline solution derived by the optimal offline algorithm designed in Sec. V-A, with complete knowledge of the system over the entire time span.

Fig. 6 shows the excessive costs against that of our lookahead algorithm at each time incurred by the simple CDN algorithm, the smart CDN algorithm and the one-shot optimum algorithm, respectively. We can see that our algorithm performs significantly better than both the simple CDN and the smart CDN algorithms, with the latter incurring much more cost due to the request dispatching heuristic applied: the smart CDN algorithm focuses on locality awareness, where each request is routed to the closest available cloud site, even though serving a request there may be more expensive than in other cloud sites. The cost incurred by the one-shot optimal solution is much less, as compared to the former two, but is still higher than the lookahead algorithm, verifying the effectiveness of the online adjustment mechanism.

Fig. 7 shows that the operational cost achieved by our algorithm is very close to the offline optimum over 24 hours, with a gap of approximately 8%. It is interesting to see that the offline optimum algorithm incurs higher cost at the beginning, due to content prefetch for future request serving.

Fig. 8 shows that the smart CDN algorithm achieves the lowest response latencies, and the other three algorithms achieve similar latencies and all meet the service quality target, *i.e.*, $150ms$.

E. Simulation at Larger Scales

Due to the prohibitive traffic cost for running experiments on Amazon EC2, we further evaluate our algorithms using large-scale simulations, to examine their performance with the

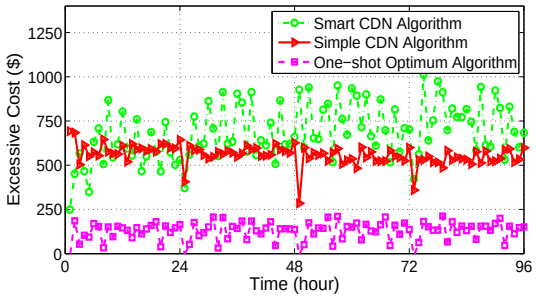

Fig. 6. Excessive operational cost against the lookahead algorithm.

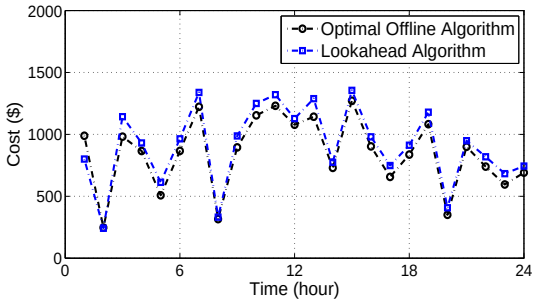

Fig. 7. Operational cost comparison with the offline optimum.

increase of the system scale. Since our algorithms only deal with the aggregate number of user requests per region, the influence of the increasing number of users on the algorithm performance is limited. We therefore only show the simulation results when the number of videos increases in the system, while fixing the total number of users at $1,000,000$. Fig. 9 plots the excessive cost percentages of all four algorithms against the offline optimum. The excessive cost percentage of an algorithm (*i.e.*, the Lookahead Algorithm, the One-shot Optimum Algorithm, the Simple CDN Algorithm, the Smart CDN Algorithm) is computed as follows, where the cost is the overall cost incurred by an algorithm in the entire (same) simulation span:

$$\frac{\text{cost of the algorithm} - \text{offline optimum cost}}{\text{offline optimum cost}}.$$

We can see that the excessive cost percentages of all four algorithms are relatively stable as the number of videos grows, and the cost incurred by the lookahead algorithm is always closest to that of the offline optimum.

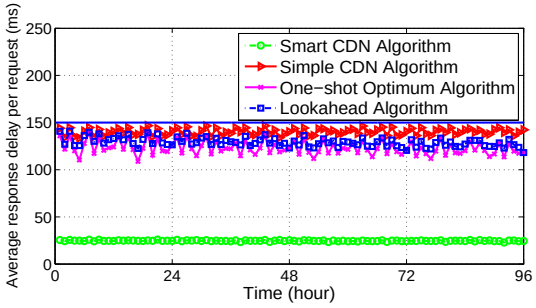

Fig. 8. Average response delay comparison.

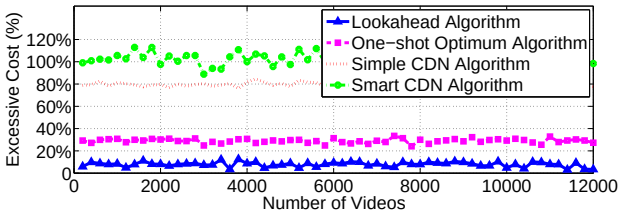

Fig. 9. Excessive operational cost against the offline optimum: a simulation study with increasing numbers of videos.

VII. Concluding Remarks

This paper introduces a proactive, online algorithm to scale social media streaming applications for operating in geo-distributed clouds. Exploiting the underlying social influences among the users, we build a simple, effective epidemic model to predict future viewing demands for proactive service deployment. Aiming at operational cost minimization with service delay guarantees, we formulate an optimal content migration and request distribution problem, with long-time and one-shot flavors, respectively. Efficient methods are proposed to solve the one-shot optimization, and a novel $\Delta(t)$-step look-ahead mechanism is designed with guarantees to adjust the one-shot optimum to the offline optimum, which is based on solid theoretical analysis. Our large-scale evaluations on an emulated distributed cloud over the Amazon EC2 platform under realistic settings confirm the excellent performance of our online algorithm in pursuing the ultimate optimal replication and request dispatching solutions, using limited information within small look-ahead windows.

Appendix A
Proof of Lemma 1

Proof: For the relaxed linear program of subproblem (A), the optimal solution(s) can only be the vertex (vertices) of the polyhedron formed by the constraints:

$$\begin{cases} -\sum_{f\in F} y_f^{(c)}(t) \le -1, \forall c \in C(t) \\ y_f^{(c)}(t) \le 1, \forall c \in C(t), \forall f \in F \\ -y_f^{(c)}(t) \le 0, \forall c \in C(t), \forall f \in F \end{cases} \tag{7}$$

Let M denote the constraint matrix:

M = $\begin{pmatrix} M_1{}_{(|C(t)|)\times(|C(t)|\times|F|)} \\ M_2{}_{(|C(t)|\times|F|)\times(|C(t)|\times|F|)} \\ M_3{}_{(|C(t)|\times|F|)\times(|C(t)|\times|F|)} \end{pmatrix}$,

where M_1 = $\left(\begin{array}{ccc|ccc|c|ccc} -1 & \cdots & 0 & -1 & \cdots & 0 & \cdots & -1 & \cdots & 0 \\ & \ddots & \vdots & & \ddots & \vdots & \ddots & & \ddots & \vdots \\ & & -1 & & & -1 & -1 & & & -1 \end{array}\right)$,

$M_2 = E$, and $M_3 = -E$, where E is the unit matrix. Let $b = (-1, -1, \ldots, -1, 1, 1, \ldots, 1, 0, 0, \ldots, 0)^T$, which is a $|C(t)|+|C(t)|\times|F|+|C(t)|\times|F|$-dimension vector.

If M is *totally unimodular*, then every vertex of the polyhedron formed by $M\vec{y} \le b$ is integral. So we can prove Lemma 1 if we can show that M is *totally unimodular*.

For any 1-by-1 sub matrix of M, we know the determinant can only be -1, +1 or 0, since the entry of M is in $\{$+1, -1, 0$\}$. By inductive hypothesis, we assume that the possible determinant of any square sub matrix of M with a dimension of no greater than $N \times N$ is in $\{$-1, +1, 0$\}$. We will prove that the determinant of any sub matrix $M'_{(N+1)\times(N+1)}$ can only be +1, -1 or 0.

For sub matrix $M'_{(N+1)\times(N+1)}$, there are two cases:

(i) There is a row r in $M'_{(N+1)\times(N+1)}$ which is (part of) a row in M_3. We can easily see that, the corresponding row in $M'_{(N+1)\times(N+1)}$ has at most one non-zero entry, *i.e.*, -1. Denote M'_r as the corresponding row in $M'_{(N+1)\times(N+1)}$. Therefore,

$$det(M'_{(N+1)\times(N+1)}) = \begin{cases} 0, & if \quad M'_r = \vec{0} \\ \{+1,-1\} \times det(M'_{N\times N}) = \{+1,-1,0\}, & otherwise \end{cases}$$

(ii) $M'_{(N+1)\times(N+1)}$ consists only of rows from

$$\begin{pmatrix} \underbrace{M_1}_{(\lfloor C(t)\rfloor)\times(\lfloor C(t)\rfloor\times\lfloor F\rfloor)} \\ \underbrace{M_2}_{(|C(t)|\times|F|)\times(|C(t)|\times|F|)} \end{pmatrix}.$$

The above $(|C(t)| + |C(t)||F|) \times (|C(t)||F|)$-dimensional matrix is *totally unimodular* according to the *Ghouila-Houri's characterization* [43], since each column has exactly one 1 and -1. Then based on the definition of total unimodularity [34], we know that the determinant of every square submatrix of the above matrix (consisting of M_1 and M_2) is in $\{$-1, +1, 0$\}$. Therefore, since $M'_{(N+1)\times(N+1)}$ is a square submatrix of the above matrix, its determinant is in $\{$-1, +1, 0$\}$.

Therefore, we have proven that the determinant of any sub matrix of M is in $\{$+1, -1, 0$\}$, and thus M is *totally unimodular* and the lemma is proven.

■

Appendix B
Proof of Theorem 1

Proof: In each time slot t, , the number of replication strategies for each content is $\binom{|F|}{0} + \binom{|F|}{1} + \binom{|F|}{2} + ... + \binom{|F|}{|F|}$, and the number of replication strategies for all contents in t is

$$|\mathcal{P}(t)| = \left(\binom{|F|}{0} + \binom{|F|}{1} + \binom{|F|}{2} + ... + \binom{|F|}{|F|}\right)^{|C(t)|} = (2^{|F|})^{|C(t)|}.$$

In any given time slot t, the number of all $Opt(t, \vec{y}(t))$'s to compute is $|\mathcal{P}(t)|$, each at a given $\vec{y}(t) \in \mathcal{P}(t)$. To compute each $Opt(t, \vec{y}(t))$ at a given $\vec{y}(t)$ using (6), we can loop through $|\mathcal{P}(t-1)|$ possible choices of $\vec{y}(t-1) \in \mathcal{P}(t-1)$, and at each fixed $\vec{y}(t-1)$, we solve a linear minimization problem as follows,

$$\min_{\vec{\alpha}(t): \text{(b)-(f) in (1)}} \mathbb{F}(\vec{y}(t), \vec{\alpha}(t)), \tag{8}$$

to derive $\vec{\alpha}(t)$ where $\vec{y}(t)$ is given. (8) is the one-shot optimization problem in (2) to compute the request dispatching strategies $\vec{\alpha}(t)$ with content replication strategies $\vec{y}(t)$ given.

Therefore, if we take the derivation of the optimal request dispatching strategy $\vec{\alpha}^*(t)$ based on the one-shot optimization problem in (2) with given replication decision $\vec{y}^*(t)$ as one atomic operation (*i.e.*, solving (8) as one atomic operation), the computation complexity to calculate all $Opt(t, \vec{y}(t))$'s in a time slot t is at most $O((\max_{\tau\in[1,T]} |\mathcal{P}(\tau)|)^2)$, *i.e.*, the number of $Opt(t, \vec{y}(t))$'s to compute is at most $\max_{\tau\in[1,T]} |\mathcal{P}(\tau)|$ and each $Opt(t, \vec{y}(t))$ is computed by looping through at most $\max_{\tau\in[1,T]} |\mathcal{P}(\tau)|$ possible choices of $\vec{y}(t-1)$. Given the total number of time slots is T, the overall computational complexity is:

$$O(T(\max_{\tau\in[1,T]} |\mathcal{P}(\tau)|)^2) = O(T((2^{|F|})^{\max_{\tau\in[1,T]} |C(\tau)|})^2) = O(T2^{2|F|\times\max_{\tau\in[1,T]} |C(\tau)|}).$$

■

Appendix C
Proof of Lemma 2

Proof: We prove Lemma 2 by contradiction. Let $\vec{y}^*$ denote the offline optimal solution and assume $y_f^{*(c)}(t) = 0$. We create another feasible solution $\vec{y}^{**}$ of the offline optimization in (1) by changing $y_f^{*(c)}(t)$ from 0 to 1, *i.e.*, the one-shot optimum derived by solving one-shot optimization (2) at t, while keeping all other caching decision variables to be the same values as those in $\vec{y}^*$.

We compare the overall operational cost over infinite time when $\vec{y}^{**}$ and $\vec{y}^*$ are applied as the caching strategies over time, respectively. Let $\vec{\alpha}^{**}$ and $\vec{\alpha}^*$ be the corresponding request distribution decisions, derived by solving the one-shot optimization at each time with given caching strategies $\vec{y}^{**}$ and $\vec{y}^*$, respectively. Let notation $\mathbb{F}(\vec{y}(t+1), \vec{\alpha}(t+1))|_{y_f^{(c)}(t)=x}$ denote the operational cost incurred in $t+1$ in the entire system, given $y_f^{(c)}(t) = x$, where $x = 1$ or 0 denotes video c is cached or not in region f in the previous time slot t. The overall cost difference is:

$$\begin{aligned} &\mathbb{H}(\vec{y}^{**}, \vec{\alpha}^{**}) - \mathbb{H}(\vec{y}^*, \vec{\alpha}^*) \\ &= \mathbb{F}(\vec{y}^{**}(t), \vec{\alpha}^{**}(t)) - \mathbb{F}(\vec{y}^*(t), \vec{\alpha}^*(t)) \\ &\quad + \mathbb{F}(\vec{y}^*(t+1), \vec{\alpha}^*(t+1))|_{y_f^{(c)}(t)=1} \\ &\quad - \mathbb{F}(\vec{y}^*(t+1), \vec{\alpha}^*(t+1))|_{y_f^{(c)}(t)=0} \\ &= \mathbb{F}(\vec{y}^{**}(t), \vec{\alpha}^{**}(t)) - \mathbb{F}(\vec{y}^*(t), \vec{\alpha}^*(t)) \\ &\quad - y_f^{*(c)}(t+1) \times \varphi_f \\ &\leq \mathbb{F}(\vec{y}^{**}(t), \vec{\alpha}^{**}(t)) - \mathbb{F}(\vec{y}^*(t), \vec{\alpha}^*(t)) \leq 0. \end{aligned}$$

In the above, the first equality holds because the operational cost when applying the two different strategies only differs at time slot t and $t+1$. The reason is that decisions made in the past (before time t) can not be altered. The second equality is because the operational cost difference at $t+1$ only lies in the migration cost, *i.e.*, a migration cost may occur if $y_f^{(c)}(t) = 0$, but not if $y_f^{(c)}(t) = 1$. The last inequality holds because $\vec{y}^{**}(t)$ and the corresponding $\vec{\alpha}^{**}(t)$ constitute the one-shot optimal solution for time slot t.

The result shows that $\vec{y}^{**}$ leads to smaller overall cost, which contradicts our assumption that $\vec{y}^*$ is offline optimum.

■

APPENDIX D
PROOF OF LEMMA 3

Proof: We only need to show that $y_f^{(c)}(t) = 1$ is the one-shot optimum at t when $y_f^{(c)}(t-1) = 1$. We prove it by contradiction by assuming that the one-shot optimum $y_f^{(c)}(t) = 0$ when $y_f^{(c)}(t-1) = 1$.

Suppose $\vec{y}^{'}(t)$ is the one-shot optimal solution including $y_f^{'(c)}(t) = 0$. We create another feasible solution $\vec{y}^{''}(t)$ by changing $y_f^{(c)}(t)$ from 0 to 1, while keeping all other caching decision variables to be the same values as those in $\vec{y}^{'}(t)$. Then we compare the one-shot operational cost at t when $\vec{y}^{''}(t)$ and $\vec{y}^{'}(t)$ are applied as the caching strategies, respectively. Let $\vec{\alpha}^{''}(t)$ and $\vec{\alpha}'(t)$ be the corresponding request distribution decisions, derived by solving the one-shot optimization (2) with given caching strategies $\vec{y}^{''}(t)$ and $\vec{y}^{'}(t)$, respectively. Let notation $\mathbb{F}(\vec{y}(t), \vec{\alpha}(t))|_{y_f^{(c)}(t-1)=x}$ denote the operational cost incurred in t in the entire system, if $y_f^{(c)}(t-1) = x$, where $x = 1$ or 0 denotes video c is cached or not in region f in the previous time slot $t-1$.

$$
\begin{aligned}
&\mathbb{F}(\vec{y}^{''}(t), \vec{\alpha}^{''}(t))|_{y_f^{(c)}(t-1)=1} - \mathbb{F}(\vec{y}^{'}(t), \vec{\alpha}^{'}(t))|_{y_f^{(c)}(t-1)=1} \\
&= \mathbb{F}(\vec{y}^{''}(t), \vec{\alpha}^{''}(t))|_{y_f^{(c)}(t-1)=0} - y_f^{(c)}(t) \times \varphi_f \\
&\quad -\mathbb{F}(\vec{y}^{'}(t), \vec{\alpha}^{'}(t))|_{y_f^{(c)}(t-1)=1} \\
&= \mathbb{F}(\vec{y}^{''}(t), \vec{\alpha}^{''}(t))|_{y_f^{(c)}(t-1)=0} - y_f^{(c)}(t) \times \varphi_f \\
&\quad -\mathbb{F}(\vec{y}^{'}(t), \vec{\alpha}^{'}(t))|_{y_f^{(c)}(t-1)=0} \\
&< \mathbb{F}(\vec{y}^{''}(t), \vec{\alpha}^{''}(t))|_{y_f^{(c)}(t-1)=0} - \mathbb{F}(\vec{y}^{'}(t), \vec{\alpha}^{'}(t))|_{y_f^{(c)}(t-1)=0} \\
&\leq 0.
\end{aligned}
$$

In the above, the first equality holds because a migration cost may be occurred in t if $y_f^{(c)}(t-1) = 0$, *i.e.*, video c is not stored in f in the prior time slot. The second equality is derived based on the fact that, if the one-shot optimal replication decision at t is not to store a video c in f, c's storage status in f in the prior time slot $t-1$ has no effect on the operational cost at t. The last inequality is due to our assumption that $y_f^{(c)}(t) = 1$ is the one-shot optimal solution if $y_f^{(c)}(t-1) = 0$.

The result shows that $\vec{y}^{''}(t)$ leads to smaller operational cost in t given $y_f^{(c)}(t-1) = 1$, which contradicts our assumption that $\vec{y}^{'}(t)$ is the one-shot optimum in t when $y_f^{(c)}(t-1) = 1$. ■

APPENDIX E
PROOF OF THEOREM 2

Proof: According to Lemma 2, if to cache video c in region f at t is the one-shot optimal replication decision, *i.e.*, $y_f^{(c)}(t) = 1$, for any $c \in C(t), f \in F$, it is already offline optimal, *i.e.*, $y_f^{*(c)}(t) = 1$. Thus adjustment will only be attempted if it is one-shot optimal not to cache video c in f in t, *i.e.*, $y_f^{(c)}(t) = 0$, as done by Algorithm 1.

Algorithm 1 only adjusts a $y_f^{(c)}(t)$ from 0 to 1, if (i) the two sequences of $\vec{y}_f^{0(c)}[t, \Delta(t)]$ and $\vec{y}_f^{1(c)}[t, \Delta(t)]$ merge at some $\Delta(t) \leq W_{thresh}$ steps after t, *i.e.*, $y_f^{0(c)}(t + \Delta(t)) = y_f^{1(c)}(t + \Delta(t))$, (ii) the cost incurred after the adjustment is smaller, *i.e.*, $\mathbb{G}(\vec{y}^{1(c)}[t, \Delta(t)], \vec{y}^{0(c)}[t, \Delta(t)]) < 0$, as well as that (iii) the adjustment does not affect video c's replication decisions in regions other than f during $[t, \Delta(t)]$. In this case, since the two caching decision sequences become the same again from $t + \Delta(t)$ onwards and no other replication of c is affected by the adjustment, the difference of overall cost over infinite time after and before the adjustment, is exactly $\mathbb{G}(\vec{y}^{1(c)}[t, \Delta(t)], \vec{y}^{0(c)}[t, \Delta(t)])$. Therefore, the above three conditions guarantee that Algorithm 1 only adjusts the one-shot optimum $y_f^{(c)}(t) = 0$ to $y_f^{(c)}(t) = 1$ when the aggregate operational cost over long run of the system will decrease, which is a better solution approximating the offline optimum.

■

REFERENCES

[1] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, W. Emmerich, and F. Galan, "The RESERVOIR Model and Architecture for Open Federated Cloud Computing," *IBM Journal of Research and Development*, vol. 53, no. 4, July 2009.

[2] *Open Data Center Alliance*, http://www.opendatacenteralliance.org.

[3] *Open Cloud Computing Interface*, http://forge.ogf.org/sf/go/projects.occi-wg.

[4] *Globus Project by the University of Chicago*, http://www.cca08.org/papers/Paper20- Sotomayor.pdf.

[5] *OGSA Basic Execution Services WG*, https://forge.gridforum.org/sf/projects/ogsa-bes-wg.

[6] M. Armbrust, A. Fox, R. Grifth, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. P. A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," *Technical report, EECS, University of California, Berkeley*, 2009.

[7] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50 – 55, 2009.

[8] M. Hajjat, X. Sun, Y.-W. E. Sung, D. A. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, "Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud," in *Proc. of ACM SIGCOMM*, August 2010.

[9] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A Cost-aware Elasticity Provisioning System for the Cloud," in *Proc. of IEEE ICDCS*, June 2011.

[10] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena, "Intelligent Workload Factoring for a Hybrid Cloud Computing Model," in *Proc. of the International Workshop on Cloud Services (IWCS 2009)*, June 2009.

[11] Y. Wu, C. Wu, B. Li, X. Qiu, and F. C. Lau, "CloudMedia: When Cloud on Demand Meets Video on Demand," in *Proc. of IEEE ICDCS*, June 2011.

[12] H. Li, L. Zhong, J. Liu, B. Li, and K. Xu, "Cost-effective Partial Migration of VoD Services to Content Clouds," in *Proc. of IEEE Cloud*, June 2011.

[13] J. M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez, "The Little Engine(s) That Could: Scaling Online Social Networks," in *Proc. of ACM SIGCOMM*, August 2010.

[14] X. Cheng and J. Liu, "Load-Balanced Migration of Social Media to Content Clouds," in *Proc. of ACM NOSSDAV*, June 2011.

[15] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010*. Springer, pp. 21–23.

[16] R. Zhou, S. Khemmarat, and L. Gao, "The Impact of YouTube Recommendation System on Video Views," in *Proc. of ACM IMC*, November 2010.

[17] Z. Wang, L. Sun, C. Wu, and S. Yang, "Guiding Internet-Scale Video Service Deployment Using Microblog-based Prediction," in *Proc. of IEEE INFOCOM mini conference*, 2012.

[18] K. Lai and D. Wang, "Towards Understanding the External links of Video Sharing sites: Measurement and Analysis," in *Proc. of ACM NOSSDAV*, June 2010.

[19] S. Scellato, C. Mascolo, M. Musolesi, and J. Crowcroft, "Track globally and deliver locally: Improving content delivery networks by tracking geographic social cascades," in *WWW*, Mar 28 Apr 1 2011.

[20] S. Borst, V. Gupta, and A. Walid, "Distributed Caching Algorithms for Content Distribution Networks," in *Proc. of IEEE INFOCOM*, 2010.

[21] J. Liu and B. Li, "A QoS-based Caching and Scheduling Algorithm for Multimedia Objects," *Journal of World Wide Web*, vol. 7, no. 3, pp. 281–296, September 2004.

[22] F. Chen, K. Guo, J. Lin, and T. L. Porta, "Intra-cloud lightning: Building cdns in the cloud," in *Proc. of IEEE INFOCOM*, 2012.

[23] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. C. Lau, "Scaling social media applications into geo-distributed clouds," in *Proc. of IEEE INFOCOM*, 2012.

[24] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[25] Y. Bartal, A. Fiat, and Y. Rabani, "Competitive algorithms for distributed data management," in *In Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pp. 39–50.

[26] D. L. Black and D. D. Sleator, "Compebitive algorithms for replication and migration problems," 1989.

[27] "Intel Cloud Builders Guide: Cloud Design and Deployment on Intel Platforms," *White Paper*, 2011.

[28] *Cloud Computing Definitions*, http://csrc.nist.gov/groups/SNS/cloud-computing.

[29] *Amazon Elastic Compute Cloud*, http://aws.amazon.com/ec2/.

[30] *Amazon Simple Storage Service*, http://aws.amazon.com/s3/.

[31] R. Anderson, *Population Dynamics of Infectious Diseases: Theory and Applications*. Chapman and Hall, 1982.

[32] S. Asur, B. A. Huberman, G. Szabo, and C. Wang, "Trends in Social Media : Persistence and Decay," in *Proc. of CoRR*, 2011.

[33] S. Boyd, "Primal and Dual Decomposition," Lecture Notes, EE364b Convex Optimization, Stanford University, http://www.stanford.edu/class/ee364b/lectures.html.

[34] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

[35] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis, Third Edition*. John Wiley and Sons, Inc., 2001.

[36] Y. Huang, T. Z. J. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, Design and Analysis of a Large-Scale P2P-VoD System," in *Proc. of ACM SIGCOMM*, August 2008.

[37] C. Wu, B. Li, and S. Zhao, "Multi-channel Live P2P Streaming: Refocusing on Servers," in *Proc. of IEEE INFOCOM*, 2008.

[38] R. Kuschnig, I. Kofler, and H. Hellwagner, "Improving Internet Video Streaming Performance by Parallel TCP-based Request-Response Streams," in *Proc. of CCNC*, January 2010.

[39] B. Ribeiro, W. Gauvin, B. Liu, and D. Towsley, "On MySpace Account Spans and Double Pareto-Like Distribution of Friends," in *Proc. of IEEE INFOCOM*, March 2010.

[40] H. Li, H. Wang, and J. Liu, "Video sharing in online social networks: Measurement and analysis," in *Proceedings of ACM NOSSDAV*, 2012.

[41] Z. Wang, L. Sun, X. Chen, W. Zhu, J. Liu, M. Chen, and S. Yang, "Propagation-based social-aware replication for social video contents," in *Proceedings of the 20th ACM international conference on Multimedia*, ser. MM '12. New York, NY, USA: ACM, 2012, pp. 29–38. [Online]. Available: http://doi.acm.org/10.1145/2393347.2393359

[42] G. Box, G. M. Jenkins, and G. Reinsel, *Time Series Analysis: Forecasting & Control (3rd Edition)*, 3rd ed. Prentice Hall, Feb. 1994.

[43] A. Ghouila-Houri, "Charactrisations des Matrices Totalement Unimodulaires," *Comptes Rendus de I' Acadmie des Sciences*, pp. 1192 – 1193, 1962.