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 algorithm using solid theoretical analysis, as well as large-scale experiments under dynamic realistic settings on a home-built cloud platform.**

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 limited 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 that is most cost-effective. 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 response delays and the operational costs are minimized? 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 time.

Our work proposes such an online algorithm for dynamic,

The research was supported in part by a grant from RGC under the contract HKU 718710E, a grant from NSFC/RGC under the contract N_HKUST610/11, by grants from HKUST under the contract RPC11EG29, SRFI11EG17-C and SBI09/10.EG01-C, by a grant from Huawei Technologies Co. Ltd. under the contract HUAW18-15L0181011/PN, and by a grant from Guangdong Bureau of Science and Technology under the contract GDST11EG06.

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 the 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.

Finally, performance of our algorithm is evaluated via large-scale experiments under dynamic realistic settings conducted on a home-built cloud platform. The results show that using our online algorithm, high-performance social media applications can be effectively supported by a geo-distributed cloud with minimum operational cost.

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 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. The open data center alliance [2], for instance, aims to provide solutions to unify cloud resources from different providers to produce a global scale cloud platform. The current literature focus on designing inter-connecting standards and APIs [1] [3] [4], while our study here 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*. [5], Sharma *et al*. [6], and Zhang *et al*. [7] advocate migrating enterprise IT applications to exploit the computation and storage capacities of a cloud. Wu *et al*. [8] and Li *et al*. [9] discuss migration of VoD services onto a cloud platform, by exploring demands and user patterns in a conventional VoD application. Pujol *et al*. [10] and Xu *et al*. [11] 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.

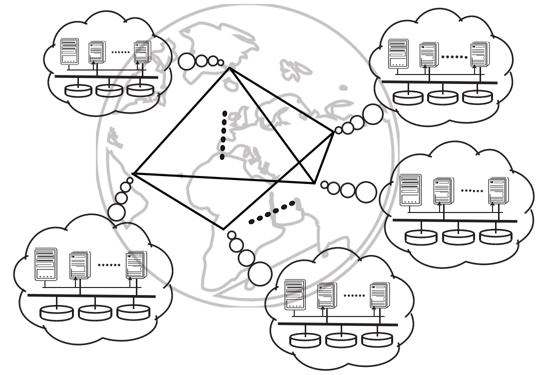

Fig. 1. The geo-distributed cloud model.

A substantial body of literature has been devoted to content replication and scheduling in a CDN or cache network [12][13]. 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.

III. System Model

A. The Geo-distributed Cloud

We consider a geo-distributed cloud infrastructure 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 [14]: 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 [15].

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 [16] and S3[17].

The cloud architecture is illustrated in Fig. 1.

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 [18]).

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 can be or 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, and let $d_f^{(c)}(t) = |D_f^{(c)}(t)|$ be the number of requests for video c from region f at 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 $v_f = \frac{m_f}{n_f} + b_f$ 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 $t = 1, \ldots, T$)

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

Constraints (a) and (b) indicate that at least one copy of each video should be stored in the cloud at any time. Constraints (c), (f) and (g) guarantee that requests would only be dispatched to a cloud site that stores the required video. In constraint (d), r_{jf} represents the round-trip delay between region j and region f (reflecting proximity in between), and $\mathcal{R}$ 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. (e) 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

[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.

[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.

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
$d_f^{(c)}(t)$	The number of users in $D_f^{(c)}(t)$
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
$\mathbb{H}(\vec{y},\vec{\alpha})$	Objective function of the offline optimization problem in (1)
$\mathbb{F}(\vec{y}(t), \vec{\alpha}(t))$	Objective function of the one-shot optimization problem in (2) at time slot t
$t_0^{(c)}$	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	Initial popularity of a video
γ_c	Decreasing speed of video c' popularity

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. We seek to design an online algorithm that pursues this optimal solution (referred to as *offline optimal 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.

IV. ONE-SHOT OPTIMIZATION

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

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

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 [19].

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[3]. $L^{(c)}(t)$ 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 [9] [20]. We employ an exponential decreasing model to describe this phenomenon: we use $\eta \times \gamma_c^{(t-t_0^{(c)})}$ to represent the probability that a potential viewer of video c may actually watch the video at t, where factor $\eta \in [0, 1]$ and $\gamma_c \in [0, 1]$ correspond to the initial value and the decreasing speed of video c's popularity, respectively.

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 $t > t_0^{(c)}$:

$$\begin{cases} s^{(c)}(t_0^{(c)}) = 0, \\ d^{(c)}(t_0^{(c)}) = 0, \\ A^{(c)}(t_0^{(c)}) = \Omega \backslash \{o^{(c)}\}, \\ s^{(c)}(t) = s^{(c)}(t-1) - d^{(c)}(t-1) \\ \quad +|\cup_{i\in E^{(c)}(t-1)} (N(i) \cap A^{(c)}(t-1)) \cup L^{(c)}(t-1)|, \\ d^{(c)}(t) = s^{(c)}(t) \times \eta \times \gamma_c^{(t-t_0^{(c)})}, \\ A^{(c)}(t) = A^{(c)}(t-1) \backslash D^{(c)}(t). \end{cases} \tag{3}$$

The rationale is as follows: When video c is uploaded at $t_0^{(c)}$, no other users than $o^{(c)}$ have watched it (Eqn. 1—3 in (3)). The potential set of viewers at t is derived in Eqn. 4 by

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

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. 5 the number of actual viewers is estimated by multiplying the number of potential viewers by probability $\eta \times \gamma_c^{(t-t_0^{(c)})}$. 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 (Eqn. 6).

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. 4 and 5, 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. 4, 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 constraints (a) and (b) in (1), and $\mathbb{C}_2$ is the set defined by linear constraints (d)—(g) in (1). This optimization problem is a mixed integer program. Nevertheless, we next show that an efficient solution indeed exists through *dual decomposition* [21].

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

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

$$\begin{aligned} &g_2(\vec{\lambda}(t)) = \min \textstyle\sum_{c\in C(t)} \sum_{j\in F} \sum_{f\in F} \alpha_{jf}^{(c)}(t)(d_j^{(c)}(t) v_f + \lambda_{jf}^{(c)}) \\ &s.t. \quad \vec{\alpha}(t) \in \mathbb{C}_2. \end{aligned}$$
(B)

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

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 can be solved by the subgradient algorithm [21], 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 [22]. 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 resulting linear program is the optimal solution to the integer program in Lemma 1.

Lemma 1. *The optimal solution to the relaxed linear program of integer subproblem (A) is integral,* i.e.*, the optimal solution of the relaxed linear program is the optimal solution to the integer program.*

The lemma is proved by showing the total unimodularity [22] of the constraint matrix of integer program (A). Readers are referred to our technical report [23] for detailed proof.

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 *offline optimal 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 next explore dependencies among video replication decisions across consecutive time slots, and design an online algorithm to improve solutions towards offline optimum.

A. Pursuing Offline Optimality with $\Delta(t)$-step Look-ahead

At $t-1$, 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 offline optimal 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, $y_f^{*(c)}(t) = 1$. Detailed proof can be found in [23].

(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[4] 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 [23]. 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

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

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

[4]The prediction can be done following our epidemic model in (3), or using other regression techniques [24].

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

$y_f^{0(c)}(t+\delta t) = 1$ 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 1 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, with detailed proof given in [23].

Theorem 1. *Given the number of viewing requests within the next $\Delta(t)$ time slots that are predicted/learned, Algorithm 1 adjusts the one-shot optimal solution at time slot t towards the offline optimum.*

B. Practical Implementation

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.

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.

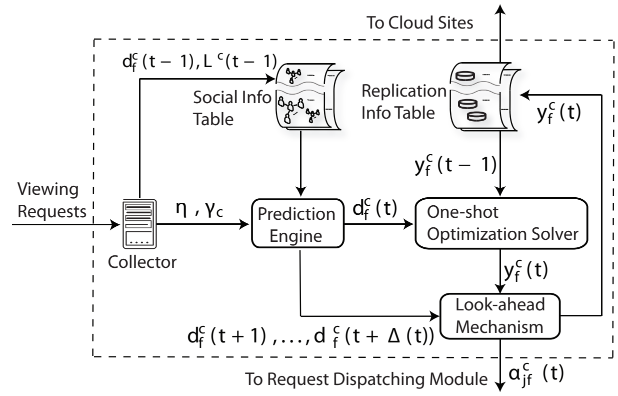

Fig. 2. Key modules in online algorithm implementation.

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 [16], 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 [25] [26]), our algorithm provides nice guidelines for optimal pre-deployment of videos.

VI. Performance Evaluation

We evaluate the performance of our online algorithm, using a home-built cloud system under realistic settings.

A. Prototype Implementation and Experimental Settings

We have built a cloud infrastructure using 50+ commodity computers (Intel(R) Pentium(R) 4 CPU 2.80GHz, 1G RAM, and 80G hard drive), interconnected via IBM 8275 Ethernet switches. On this platform, we emulate a geo-distributed cloud across 10 regions (San Francisco, New Orleans, Ottawa, Rio de Janerio, London, Berlin, Bengaluru, Singapore, Hong Kong, and Tokyo), with one data center in each. The round-trip delays (RTT) between each pair of cloud sites are emulated using manually injected delays in programs following the formula $RTT(ms) = 0.02 \times Distance(km) + 5$ from [27]. Computers belonging to one cloud site are divided into computing servers and storage servers, installed with Xen VMs and NFS file systems, respectively. Different charges are applied in the 10 cloud sites, as given in Table III. The prices are set based on the charging model of Amazon Web Services[16][17], with minor adjustments.

10 additional PCs are used to simulate user groups located in different regions, which produce viewing requests to dispatch to the cloud sites. The RTT between a user and a cloud site is 20ms if they are in the same region, or calculated using the above formula 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 [28]. Another computer is set up as the tracker server, implementing the *Collector*, *Prediction Engine*, *One-Shot Optimization Solver*, and *Look-ahead Mechanism* discussed in Sec. V.

In our experiments, there are $10,000$ users in the application. The number of friends of each user follows a lognormal distribution [29] with an average of $10, 80\%$ of which are from the same region where the user resides. Each video is 100M-byte long. Initially there are 60 videos in the system. Then in each round of our online algorithm (1 hour), two new videos will be uploaded by two randomly picked users located in an 'active' region — where the local time is between 9am and 9pm in a day. We emulate evolution of the popularity of each video by setting η around 0.5 and γ_c chosen in $[0.9, 0.99999]$. The videos are evenly divided into four types, and each video is recommended by the system to 50 users in each hour, who have recently watched a video of the same type. Each viewer of a video will immediately comment on the video, in our system. We emulate the running of the system over 100 hours.

B. Prediction accuracy

We first investigate effectiveness of our epidemic model for forecasting future viewing demands. In Fig. 3, the solid curve plots the actual viewing request number for a video in our system over an 18-hour span, which replays YouTube measurements from the literature [11]. The dotted curves correspond to predicted demands: the red curve represents the prediction done at $t = 0$ for the next 6 time slots; similarly, the other two curves correspond to prediction done at $t = 6$ and $t = 12$, for the next 6 hours, 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. In our following experiments, we will use a look-ahead window size $W_{thresh} = 2$ as the default.

C. Performance on Cost and Delay

We compare the performance of our algorithm with an algorithm using one-shot optimal solutions in each time slot, and a simple algorithm which replicates a copy of each video in each cloud site at all times. Fig. 4 shows that the overall operational cost of the system achieved with our algorithm is smaller than those incurred by the other two algorithms, verifying the effectiveness of our look-ahead mechanism. In Fig. 5, we observe that all three algorithms achieve similar satisfactory response latencies (below 150 ms) at all times, which fulfils our service quality target.

D. Impact of Look-ahead Window Size

We also investigate the performance improvement when different look-ahead window sizes are employed in our online algorithm. Fig. 6 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. 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 very similar cost saving. All these promise that a small look-ahead window is enough to achieve good cost savings in realistic environments.

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 under realistic settings confirm the superiority of our online algorithm in pursuing the ultimate optimal replication and request dispatching solutions, using limited information within small look-ahead windows.

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.

TABLE III
CONFIGURATIONS OF 10 GEO-DISTRIBUTED CLOUD SITES

	San Francisco	New Orleans	Ottawa	Rio de Janerio	London	Berlin	Bengaluru	Singapore	Hong Kong	Tokyo
φ_f (\$ per video)	6.66	7.44	7.20	7.80	7.50	7.11	7.74	6.96	7.53	6.78
p_f (\$ per byte per hour)	0.599	0.559	0.574	0.620	0.562	0.580	0.598	0.576	0.613	0.559
v_f (\$ per request)	0.038	0.035	0.038	0.040	0.039	0.038	0.035	0.034	0.037	0.035
$\mathcal{U}_f$ (requests per hour)	8,800	7,300	9,100	9,400	8,100	8,000	7,800	87,00	7,600	100

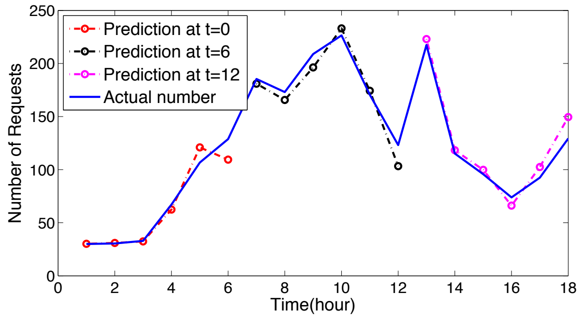

Fig. 3. Evolution of popularity of a sample video.

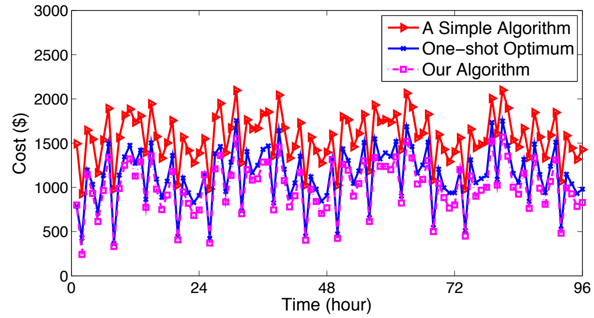

Fig. 4. Evolution of operational cost: a comparison.

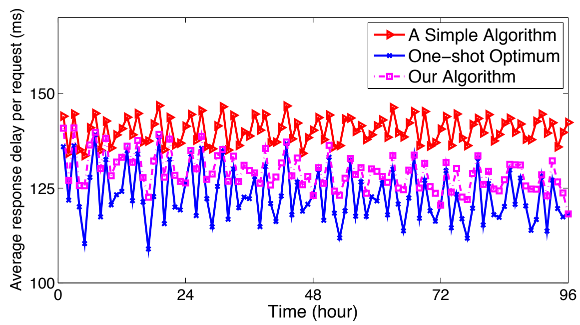

Fig. 5. Evolution of average response delay: a comparison.

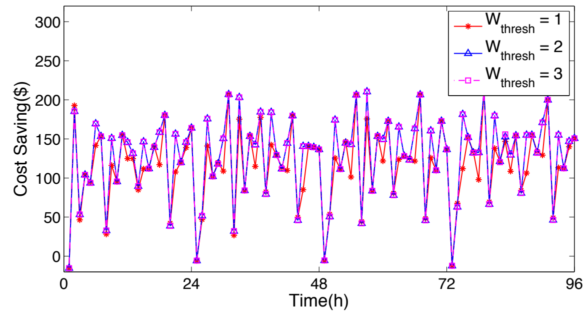

Fig. 6. Evolution of operational cost: different window sizes.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

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

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

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

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

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

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

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

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

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

[23] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. C. Lau, "Scaling Social Media Applications into Geo-Distributed Clouds," Tech. Rep., `http://www.cs.hku.hk/~cwu/papers/scalingsocial.pdf` 2011.

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

[25] 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.

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

[27] "Power-Demand Routing in Massive Geo-Distributed Systems," *PhD Thesis submitted to MIT*, 2010.

[28] 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.

[29] 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.