Meeting Service Level Agreement Cost-Effectively for Video-on-Demand Applications in the Cloud

Yuhong Zhao
Huazhong University of Science and Technology
Wuhan, China

Hong Jiang
University of Nebraska-Lincoln
Lincoln County, Nebraska
USA

Ke Zhou*, Zhijie Huang
Huazhong University of Science and Technology
Wuhan, China
k.zhou@hust.edu.cn

Ping Huang
Virginia Commonwealth University
Richmond, Virginia, USA

***Abstract*—Increasingly video-on-demand (VoD) applications have been ported to cloud platforms. Leveraging the elastic resource provisioning of the cloud, it is believed that VoD applications should attain high performance cost-effectively. In this paper we propose an approach that aims to solve the fundamental resource reservation and scheduling problem of configuring the cloud utility to meet SLAs for VoD applications at a modest cost. First, we devise a constraint-based model that describes the relationship among channel placement, user groups' bandwidth allocation, operating costs and QoS constraints. Second, we present a distributed heuristic algorithm, called DREAM, that solves the model and produces a budget solution that reserves and allocates cloud bandwidth, and determines the channel layout among datacenters. Simulations driven by data traces collected from a commercial VoD system demonstrate that DREAM provides much better access locality and data availability than and comparable streaming quality to state-of-the-art solutions at lower cloud operating costs.**

I. Introduction

During the past decade, video on demand (VOD) has become one of the most popular services on the Internet [3][4]. A successful Internet VoD system usually contains tens of thousands of videos and attracts millions of users who concurrently watch popular videos online at peak times. In general, commercial VoD deployments use either CDNs (Content Delivery Networks) or P2P (Peer-to-Peer) networks [16]. While the former achieve high availability, the latter have lower deployment cost and are more scalable [8].

With the prevalence of the cloud computing and storage platforms, many VoD applications have been ported to cloud infrastructures completely or partially [1][2][6][7]. The cloud platform is based on a "pay-as-you-go" model that enables users from anywhere at any time to access a shared pool of configurable bandwidth and storage resources, which can be on-demand provisioned and released with minimal management effort. The on-demand resource supply in the cloud aims to meet the dynamic bandwidth and storage demands of the VoD applications in real time [6].

Despite of the remarkable advantages offered by cloud platforms, new techniques are needed to fully exploit potentials promised by these advantages so as to better port VoD applications to cloud platforms. By analyzing VoD applications and their characteristics, we believe that streaming quality, access locality and data availability must be fulfilled by these new techniques to move VoD applications to the cloud in the most cost-effective ways. The main reasons are as follows. From a user's perspective, the global users must be able to download data from their respective local regions at a rate no lower than the video playback rate in order to watch the desired video streams online smoothly after a short video start-up lag. As a result, it is necessary to ensure streaming quality and access locality for VoD applications. From a video provider's viewpoint, the data availability is of paramount importance, because any loss of video data may risk violating the service level agreement (SLA) and thus result in certain forms of penalty (e.g., of economical and/or regulatory nature). Furthermore, access locality enables cloud providers to reduce the cross-boundary traffic. On the other hand, these new techniques must provide cost effectiveness because the cloud resource comes at a cost that VoD service providers would certainly want to minimize.

It is generally believed that if VoD applications are deployed in a cloud environment, they should attain high performance, in terms of streaming quality, video availability and access locality, more economically. To achieve this goal, a number of recent studies in the literature have proposed various optimizations and scheduling schemes that can be broadly divided into two categories. One category is based on cloud servers [6][8] and the others is based on video channels [5] since a VoD application usually contains many videos that are referred to as video channels. To accommodate the dynamic demands of VoD users, the former focuses on how to adaptively adjust resource and position of cloud servers while the latter concentrates on adaptively managing the video channels and the streams from them. Although both have been shown to be effective to some extend and under certain conditions, they both have clear disadvantages that are elaborated next. For cloud-server based solutions, first, to adaptively meet users' demands and changes in resource availability in real time, a scheduling algorithm [5][8] must be executed repeatedly at a relatively short time interval (e.g., ten minutes). However, there may not be sufficient time for cloud servers to adjust due to the necessary time it takes to start a new virtual machine (VM), to copy data, and to run the optimization algorithm. Second, the lack of consideration by the cloud-server based scheduling schemes to strategically place replica videos in datacenters (DCs) significantly weakens their ability to provide data availability.

*The Corresponding Author is Ke Zhou (k.zhou@hust.edu.cn).

978-1-4799-3360-0/14/$31.00 © 2014 IEEE

While the video-channel based techniques can generally manage the use of video channels among DCs in a more flexible manner than managing at the cloud-server level in the cloud platform, we believe that what is apparently lacking in the existing video-channel based techniques is a thorough and holistic consideration in their decision-making of the relationship between the operating cost of, maintaining QoS of VoD in the cloud platform and the resource assignment. Without such a thorough consideration and understanding of the relationship these optimization techniques often fail to provide the desired QoS mentioned above in a cost-effective way. For example, the latest study in this area, by Niu et al. [5] and one that is closest to ours in this paper, proposed a predictive resource auto-scaling system based on video channels that dynamically places videos and books the minimum bandwidth resources from multiple data centers to match the users' next short-term bandwidth demand with high probability. After carefully analyzing this system [5] and the pricing schemes of existing cloud platform [9][10], we found that the three optimization algorithms used by the proposed auto-scaling system to place videos into DCs and assign bandwidth to them incur relatively high cost in terms of data transfer and storage. In addition, because of possible failures at DCs [17] and user globalization [8], data availability and access locality are becoming increasingly important. However, these proposed optimization algorithms do not provide any mechanism to maintain video availability and access locality.

To address this challenge, in this paper, we propose a holistic approach that aims to solve the fundamental resource reservation and scheduling problem of configuring the cloud utility to meet SLAs for VoD applications at a modest cost. The main ideas behind and thus contributions of our proposed approach are twofold. First, we devise a constraint-based optimization model that describes the relationship among channel placement, user groups' bandwidth allocation, total operating costs and QoS constraints, and prove that solving this model is NP-hard. Second, aiming to satisfy the SLA-specified QoS requirements of VoD streaming quality, video availability, access locality in a cost-efficient way, we present a *d*istributed heuristic *re*source scheduling *a*lgorith*m* based on video channels, called *DREAM*, which produces a budget solution that reserves and allocates cloud bandwidth, and determines the channel layout among DCs. This algorithm is then extended to incorporate locality awareness by adjusting a coefficient.

We evaluate the effectiveness of a prototype implementation of *DREAM* using real-world VoD service traces collected from Youku [12] (a large commercial VoD site and China's equivalent of YouTube), as well as real-world data transfer and storage pricing policies [9]. Compared to the existing state-of-the-art solution proposed by Niu et al. [5], the prototype evaluation results demonstrate that *DREAM* saves 9%-50% operating cost of the cloud while offering higher data availability and access locality than and comparable streaming quality to state-of-the-art solutions.

II. THE GOVERNING MODEL FOR RESOURCE SCHEDULING

In this section, we formally define the mathematical model governing resource scheduling in the context of this paper, which describes the relationship among video placement, QoS constraints, bandwidth assignment and cloud operating costs for VoD applications in the cloud.

TABLE I: NOTATIONS AND THEIR MEANINGS

Notations	Meaning
M	Total number of datacenters in the system
N	Total number of user groups
W	Total number of channels to be replicated
D_i	The ith datacenter(DC)
$conf_i$	Confidence level of DC D_i
S_i	Storage capacity of DC D_i
B_i	Bandwidth capacity of DC D_i
U_j	The jth user group
c_{ij}	Communication distance between DC D_i and user group U_j
V_k	The kth channel
v_k	Size of the channel V_k
V_k^t	The threshold of channel V_k's availability
k_i	Storage cost factor of DC D_i that describes the cost of storing a unit data per unit of time
r_{ik}	Communication distance from D_i to its nearest neighboring DC that has V_k's replica
Q_{jk}	The expected bandwidth of U_j when it downloads channel V_k
X_{ijk}	The bandwidth assigned by DC D_i to user group U_j for downloading channel V_k
Y_{ik}	If channel V_k is replicated at D_i, $Y_{ik} = 1$, otherwise $Y_{ik} = 0$
$avail(V_k)$	The availability of the channel V_k
δt	The interval between two runs of the optimization algorithm
C_{ij}	$c_{ij} * \delta t$, it expresses the price of bandwidth
K_i	$k_i * \delta t$, it expresses the cost of storing a unit data during δt
$C_{\delta t}$	The operational cost of an application in δt

A. System Architecture and Notation Definition

The most frequently used notations are listed in Table I. Without loss of generality, we assume the cloud platform to be composed of M datacenters (DCs). Let D_i, S_i and B_i be the name, the total storage capacity and bandwidth capacity, respectively, of DC i where $i = 1, \ldots, M$. These DCs are geographically distributed among different regions, and they can communicate with one another through a network (e.g. WAN).

To expose and exploit access locality, we differentiate users into groups based on their network status. A user group is composed of the users from within a given geographical region or of a certain network status (e.g., an area with a certain level of physical proximity, local area network, or Autonomous System (AS)). There are N user groups, $U_1, \ldots, U_N$, watching channels from the cloud platform. Users can access the cloud platform through the Internet. Each route between data center D_i, $i = 1, \ldots, M$ and user group U_j, $j = 1, \ldots, N$ has a positive floating-point value function c_{ij}, specifying the communication distance for transferring data units between D_i and U_j. The communication distance, determined by the locations of the DC and, sometimes, the user group, can be represented in terms of the data transfer price between two regions as in [9][10], or cost of leasing the network links. Traditionally, a cloud provider leases or owns dedicated lines connecting its DCs [13], so in this paper we assume that the inter-datacenter network of a cloud provider and the network connecting the cloud platform and users are disjoint and thus independent of each other. On the other hand, if the cloud platform uses the same network to transmit data among its DCs and between users and DCs, our method can also solve the problem by simply considering a DC as a special user group.

We use time-series techniques [5] to predict every user group's future bandwidth demands for each channel. To cope with the practical challenges of (1) new video channels' lack of sufficient demand history and (2) unpredictability of some new and small channels with only a few online users, we aggregate new channels and small channels into virtual new-channels and virtual small-channels respectively as defined by Niu et al. [5]. Because of the similarity among initial demand evolution patterns of all channels, it is feasible to predict demands for new videos based on the trace of earlier videos. And a virtual new-channel including many new videos which is considered a single entity has enough online users to predict its bandwidth demand. Similarly, small video channels with few online users are aggregated into virtual small-channels. Besides these two types of virtual channels, the rest of channels are referred to as mature channels. So a virtual channel may include many videos and a mature channel includes only one video.

We assume that there are W channels, including all virtual new-channels, virtual small-channels and mature channels. Each channel V_k with a size of v_k , $k = 1, \ldots, W$, will be stored in the cloud, and our scheme will appropriately distribute its replicas among DCs. We use $k_i * v_k$ as V_k's storage cost in D_i during a unit time, in which k_i is the storage cost factor depending on the power consumption, equipment price and so on. Generally the cost of replicating a channel from one DC to another is proportional to the channel's size. We refer to this cost as migration cost and use $r_{ik} * v_k$ to calculate it, where r_{ik} is the communication distance from D_i to its "nearest" neighboring DC that has channel V_k's replica.

In this paper if the users obtain sufficient bandwidth, we think that the streaming quality is provided desirably. We denote by Q_{jk} the amount of bandwidth that should be reserved from all DCs during the next interval δt for user group U_j watching channel V_k, and use a random variable L_{jk} as the actual aggregate bandwidth load imposed on V_k from U_j during δt. To maintain the streaming quality, the load imposed on V_k from U_j should not be more than the reserved bandwidth Q_{jk} with high probability, i.e. $Pr(L_{jk} > Q_{jk}) \leq \varepsilon$, where $\varepsilon > 0$ is a small constant, called the under-provision probability. Niu et al. [5] have shown that each L_{jk} follows a Gaussian distribution. Based on their conclusion and using demand expectation and variance of L_{jk} as input, which are predicted by using the time-series techniques, we derive Q_{jk}, $j = 1, \ldots, N$ and $k = 1, \ldots, W$. All these Q_{jk}s, which are distributed to all DCs, define an $N * W$ demand matrix $\boldsymbol{Q}$. Now, we use $\boldsymbol{X}$ to denote the reserved bandwidth distribution matrix as follows. Each element of the $M * (N * W)$ matrix, X_{ijk}, for $i = 1, \ldots, M, j = 1, \ldots, N$ and $k = 1, \ldots, W$, is a real number representing the bandwidth reserved in DC D_i for user group U_j downloading channel V_k during the next interval δt. To maintain the streaming quality of the next interval, we must ensure that $\sum_{i=1}^{M} X_{ijk} \geq Q_{jk}$, $j = 1, \ldots, N$ and $k = 1, \ldots, W$. Let $Y_{ik} = 1$ if D_i holds a replica of channel V_k during interval δt, and 0 otherwise. Y_{ik}s define an $M * W$ replication matrix $\boldsymbol{Y}$ for $i = 1, \ldots, M$ and $k = 1, \ldots, W$, with Boolean elements.

B. Availability

The cloud platform aims to keep the data availability above a minimum level. We adopt the following definition of availability to express the probability of a video being available:

$$Pr(V_k \ available) = (1 - \prod_{i=1}^{M} F_i * Y_{ik}) \geq V_k^p \qquad (1)$$

where $F_i \in [0, 1]$ is the probability with which a video replica is unavailable in data center D_i. We assume that all of videos have diverse availability requirements, so we set different availability thresholds to them. Channel V_k's availability threshold is denoted by V_k^p. We further require that a given virtual channel only include videos with the same availability requirements. Thus, any channel's availability threshold is equal to that of its videos. Letting $conf_i = -log(F_i)$ and $V_k^t = -log(1 - V_k^p)$, we obtain an equivalent formula for ensuring channel availability:

$$avail(V_k) = Pr(V_k \ available) = \sum_{i=1}^{M} conf_i * Y_{ik} \geq V_k^t \quad (2)$$

where $conf \in [0, +\infty]$ is referred to as the confidence level of a DC that represents the probability of the DC not losing data in it, which is estimated based on its hardware components and fault-tolerant scheme. All V_k^ts, $k = 1, \ldots, W$, form a W-element availability demand vector $\boldsymbol{V^t}$.

C. Mathematical Model

One of our model's goals is to minimize the total operating cost, denoted as $C_{\delta t}$, which will be incurred by VoD applications during the next short-term δt. Since the communication cost of the control messages has minor impact to the overall performance of the system [14][18], we do not consider it in our model. There are three main components affecting the total operating cost, as described below:

$$C_{\delta t} = \sum_{k=1}^{W} \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} * X_{ijk} * \delta t + \sum_{k=1}^{W} \sum_{i=1:Y_{ik}=1}^{M} k_i * v_k * \delta t + \sum_{k=1}^{W} \sum_{i=1:Y_{ik}=1 and Y'_{ik}=0}^{M} r_{ik} * v_k. \quad (3)$$

The first component is the delivery cost that is introduced by sending data form DCs to user groups. $X_{ijk} * \delta t$ is the expected amount of data delivered by DC D_i to user group U_j during the interval δt. The second component is the cost arising from storing video providers' data. The third component is the replica migration cost due to the inter-DC movement of replicas as a result of the optimization algorithm's new decision on replicas' locations. Let $C_{ij} = c_{ij} * \delta t$ and $K_i = k_i * \delta t$ and put them into Eq. 3, so that the equivalent objective function is:

$$minC_{\delta t} = \sum_{k=1}^{W} \sum_{i=1}^{M} \sum_{j=1}^{N} C_{ij} * X_{ijk} + \sum_{k=1}^{W} \sum_{i=1:Y_{ik}=1}^{M} K_i * v_k + \sum_{k=1}^{W} \sum_{i=1:Y_{ik}=1 and Y'_{ik}=0}^{M} r_{ik} * v_k, \quad (4)$$

where Y'_{ik} is equal to one when the channel V_k is already replicated in DC D_i by the last execution of the optimization algorithm, otherwise it is equal to zero.

The proposed mathematical model discussed in this paper can be succinctly described by its objective function as: *Find the assignments of real values in the $\boldsymbol{X}$ matrix and of Bolean $\{0, 1\}$ values in the $\boldsymbol{Y}$ matrix that are related to $\boldsymbol{X}$, so that*

$C_{\delta t}$ in Eq. 4 is minimized subject to the following constraints:

$$if \sum_{j=1}^{N} X_{ijk} > 0 then Y_{ik} = 1, i = 1, \ldots, M \text{ and } k = 1, \ldots, W, \tag{5}$$

$$\sum_{k=1}^{W} \sum_{j=1}^{N} X_{ijk} \leq B_i, i = 1, \ldots, M, \tag{6}$$

$$\sum_{k=1}^{W} v_k * Y_{ik} \leq S_i, i = 1, \ldots, M, \tag{7}$$

$$X_{ijk} \geq 0, i = 1, \ldots, M,\ j = 1, \ldots, N \text{ and } k = 1, \ldots, W, \tag{8}$$

$$Y_{ik} \in \{0, 1\}, i = 1, \ldots, M \text{ and } k = 1, \ldots, W, \tag{9}$$

$$\sum_{i=1}^{M} X_{ijk} \geq Q_{jk}, j = 1, \ldots, N \text{ and } k = 1, \ldots, W, \tag{10}$$

$$\sum_{i=1}^{M} conf_i * Y_{ik} \geq V_k^t, k = 1, \ldots, W. \tag{11}$$

In Constraint 5, the inequality $\sum_{j=1}^{N} X_{ijk} > 0$ implies that some user groups need to access a specific channel V_k from DC D_i, so that the replica of V_k must be placed at D_i, i.e., Y_{ik} has to be 1. Constraints 6 and 7, which are the capacity constraints, states that the amount of bandwidth and storage reserved from data center D_i cannot exceed its capacity. Constraint 8 stipulates that the amount of the reserved bandwidth is a positive value. Constraint 9 stipulates that a replica of channel V_k is either placed or not placed in DC D_i. To guarantee the streaming quality and channel availability, we introduce Constraints 10 and 11.

D. Analysis and Transformation of the Model

Theorem 1. *The minimization problem described by Eq. 4 and Constraints 5-11 is NP-hard.*

Proof: Let's first make the following three assumptions: (1) there are no accesses from users to channels so that all X_{ijk}s are zero; (2) communication distance between any two DCs is equal to zero so that migration cost can be ignored; and (3) a single replica can provide sufficient availability for each channel. Based on these assumptions, all the terms of Eq. 4, except for the second term, and Constraints 5, 6, 8 and 10 can be removed. So the original problem is reduced to the following simplified minimization sub-problem:

$$minC_{\delta t} = \sum_{k=1}^{W} \sum_{i=1:Y_{ik}=1}^{M} K_i * v_k, \tag{12}$$

subject to Constraints 7, 9 and

$$\sum_{i=1}^{M} Y_{ik} \geq 1, k = 1, \ldots, W. \tag{13}$$

Obviously, this sub-problem is a minimization version of the generalized assignment problem, which Chekuri et al. [11] refers to as *Min GAP* and is NP-hard. Since the NP-hardness of a sub-problem implies that property for the general problem, we can conclude that the minimization problem from our model is NP-hard. ■

To remove Constraints 10 and 11, we introduce two penalty functions. If any demand for downloading bandwidth or availability cannot be met, the objective function value of Eq. 4 will be increased by a big number depended on the quantity of the missing bandwidth or availability. Therefore, to solve the minimality model, the optimization algorithm has to try its best to avoid the events of bandwidth or availability being unmet. With regard to Constraint 10, the penalty function term is $p_{jk}^1 = P_1 * (Q_{jk} - \sum_{i=1}^{M} X_{ijk})$, when $Q_{jk} > \sum_{i=1}^{M} X_{ijk}$, otherwise $p_{jk}^1 = 0$. And with regard to Constraint 11, the penalty function term is $p_k^2 = P_2 * (V_k^t - \sum_{i=1}^{M} conf_i * Y_{ik}) * v_k$, when $V_k^t > \sum_{i=1}^{M} conf_i * Y_{ik}$, otherwise $p_k^2 = 0$. As penalty factors, both P_1 and P_2 are real numbers. So we get a new equivalent model with Constraints 5-9 and a new objective function, which is:

$$minC_{\delta t} = \sum_{k=1}^{W} \sum_{i=1}^{M} \sum_{j=1}^{N} C_{ij} * X_{ijk} + \sum_{k=1}^{W} \sum_{i=1:Y_{ik}=1}^{M} K_i * v_k + \sum_{k=1}^{W} \sum_{i=1:Y_{ik}=1 and Y'_{ik}=0}^{M} r_{ik} * v_k + \sum_{k=1}^{W} \sum_{j=1}^{N} p_{jk}^1 + \sum_{k=1}^{W} p_k^2. \tag{14}$$

The above minimization problem can be translated into an equivalent maximization problem in which we maximize the overall virtual gain of resource assignment obtained by replicating the channels and allocating bandwidth. From Eq. 14 we can see that meeting some bandwidth or availability demands will introduce a decreased value in penalty and an increased value in operating cost. In general, we make the former is bigger than the latter by setting the two penalty factors and then define the virtual gain of resources assignment as difference value between them. Thus, the equivalent maximization problem is:

$$max \sum_{k=1}^{W} (q_k^2 - \sum_{i=1:Y_{ik}=1}^{M} K_i * v_k - \sum_{i=1:Y_{ik}=1 and Y'_{ik}=0}^{M} r_{ik} * v_k) + \sum_{k=1}^{W} \sum_{j=1}^{N} (q_{jk}^1 - \sum_{i=1}^{M} C_{ij} * X_{ijk}), \tag{15}$$

subject to Constraints 5-9, where $q_k^2 = P_2 * \sum_{i=1}^{M} conf_i * Y_{ik} * v_k$ when $V_k^t > \sum_{i=1}^{M} conf_i * Y_{ik}$, otherwise $q_k^2 = P_2 * V_k^t * v_k$, and $q_{jk}^1 = P_1 * \sum_{i=1}^{M} X_{ijk}$ when $Q_{jk} > \sum_{i=1}^{M} X_{ijk}$, otherwise $q_{jk}^1 = P_1 * Q_{jk}$.

The first term of Eq. 15 represents the additional gain obtained by ensuring the V_k's availability, while the second term represents the virtual gain obtained by maintaining stream quality for user group U_j watching channel V_k. Corresponding to the case that all demands are met, the most virtual gain of resource assignment is given by the difference between the value of penalty over all channels if no resources are assigned ($\sum_{k=1}^{W} \sum_{j=1}^{N} P_1 * Q_{jk} + \sum_{k=1}^{W} P_2 * V_k^t * v_k$) and the operating cost obtained by resources assignment given by the objective function in Eq. 4.

III. DISTRIBUTED RESOURCE SCHEDULING ALGORITHM

A. Preliminaries

In this section, we propose a *d*istributed heuristic *re*source scheduling *a*lgorith*m* based on video channels, called *DREAM*,

which solves the optimization model described by Eq. 15 and Constraints 5-9. The algorithm determines the placement of channel replicas and bandwidth reservation based on the ratio between the resource-assignment gain and the cost described by Eq. 15 and Eq. 4 respectively. The algorithm has as input three parameters, $\boldsymbol{Q}$, $\boldsymbol{V_t}$ and $\boldsymbol{Y_i'}$. The first parameter $\boldsymbol{Q}$ is the $N * W$ matrix of N user groups' bandwidth demands for W channels, which are derived by prediction and probability calculation. The second parameter $\boldsymbol{V_t}$ is the W-element vector $(V_1^t, V_2^t, \ldots, V_W^t)$ of which each entry is the threshold of a channel's availability. The third parameter $\boldsymbol{Y_i'}$ is the W-element vector of which each entry Y_{ik}', $k = 1, \ldots, W$, is equal to one when the channel V_k is already replicated in datacenter (DC) D_i by the last execution of *DREAM*, otherwise it is zero.

To help describe the *DREAM* algorithm we define three additional parameters, the variable matrix of resource assignment cost ($\boldsymbol{rc}$), the assignment gain ($\boldsymbol{rg}$) matrix and the cost-effectiveness ($\boldsymbol{ce}$) matrix. The elements of these matrices are defined as follows. The resource (storage space and bandwidth) allocation cost for a tuple consisting of DC D_i, user group U_j and channel V_k is defined as follows:

$$rc_{i0k} = (K_i * v_k + r_{ik} * v_k * (1 - Y_{ik}')) * (1 - Y_{ik}), \quad (16)$$

$$rc_{ijk} = C_{ij} * x_{ijk} + rc_{i0k}, \quad (17)$$

where $x_{ijk} = min(dQ_{jk}, eB_i)$, dQ_{jk} records U_j's remaining bandwidth demand for V_k and eB_i records the quantity of DC D_i's remaining bandwidth. Clearly, rc_{i0k} represents the cost of placing a channel V_k's replica in DC D_i, including storage cost and migration cost. In Eq. 16 the term $r_{ik} * v_k * (1 - Y_{ik}')$ is simply the migration cost that will be zero if V_k's replica was already placed in D_i by the last execution of the algorithm ($Y_{ik}' = 1$). rc_{ijk} represents the sum of the cost the application will experience if data center D_i provides x_{ijk} bandwidth to user group U_j for downloading the channel V_k. The first term of Eq. 17 $C_{ij} * x_{ijk}$ is the delivery cost.

Corresponding to the resource assignment costs, the resource assignment gains are defined as follows:

$$rg_{i0k} = \begin{cases} (P_2 * y_{ik} * v_k - rc_{i0k})(1 - Y_{ik}) \\ \qquad \text{if} \sum_{i=1}^{M} conf_i * Y_{ik} < V_k^t \\ -rc_{i0k} \\ \qquad \text{if} \sum_{i=1}^{M} conf_i * Y_{ik} \geq V_k^t, \end{cases} \quad (18)$$

if $\sum_{i=1}^{M} conf_i * Y_{ik} < V_k^t$, $y_{ik} = min((V_k^t - \sum_{l=1}^{M} conf_l * Y_{lk}), conf_i)$, otherwise 0,

$$rg_{ijk} = (P_1 - C_{ij}) * x_{ijk} + rg_{i0k}. \quad (19)$$

As can be seen from the definition of rg_{i0k}, it represents the increase in overall gain the VoD application would experience if data center D_i replicates channel V_k. The term $\sum_{i=1}^{M} conf_i * Y_{ik} \geq V_k^t$ in Eq. 18 means that channel V_k has sufficient number of replicas in different DCs to ensure its availability so that no additional replicas are needed, hence rg_{i0k} will not be greater than zero. Of course, we will set $P_2 * y_{ik}$ greater than the sum of K_i and r_{ik} to make rg_{i0k} a positive number when higher availability is required for the channel V_k. rg_{ijk} represents the sum of gain in the application when bandwidth x_{ijk} is allocated to users. The first term $(P_1 - C_{ij}) * x_{ijk}$ in Eq. 19 indicates the gain of allocating bandwidth. Similar to P_2, we make P_1 greater than C_{ij}.

Now we define the elements of the cost-effectiveness matrix of allocating resources, ce_{ijk}, as follows:

$$ce_{ijk} = rg_{ijk}/rc_{ijk}. \quad (20)$$

The cost-effectiveness element ce_{ijk} is used to determine each "global" decision of replicating channel and/or assigning bandwidth. In making these decisions, *DREAM* considers the effect of allocating the resources on the overall application gain and cost. According to Eq. 17, Eq. 19 and Eq. 20, some elements of cost-effectiveness matrix $\boldsymbol{ce}$ are recalculated after every resource assignment by *DREAM* and they will be non-positive if the demands are met or there are no more resources that can be assigned in corresponding DCs. *DREAM* always assigns resources that attain the global maximum cost-effectiveness value ce_{max}, until all elements in $\boldsymbol{ce}$ are non-positive.

B. The Proposed Algorithm

The proposed distributed heuristic algorithm for resource scheduling is given in *DREAM* as shown in Algorithm 1. The algorithm is executed by each DC within the cloud platform. It starts with an initialization phase (lines 2-15) in which the DCs initialize their local variables. In line 2, the algorithm initializes row i of the allocation matrices $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{ce}$ to zero, indicating that no replicas are placed and no bandwidth are reserved and assigned in DC D_i. And in line 3, it also initializes the available of storage space eS_i to S_i and the available of bandwidth capacity eB_i to B_i. S_i and B_i are the storage and bandwidth capacity of DC D_i. Matrices $\boldsymbol{dQ}$ and $\boldsymbol{dV^t}$ record all remaining bandwidth and availability demands. According to the second term of Eq. 16, in line 4 the function *Compute_mc* computes the cost of migrating a channel to a DC (mc_{ik}). From line 5 to line 13, our main purpose is to compute each ce_{ijk} in DCs, for $i = 1, \ldots, M$, $j = 1, \ldots, N$ and $k = 1, \ldots, W$. We compute the W-vector $\boldsymbol{rc_{i0}}$ and $\boldsymbol{rg_{i0}}$ in a given DC D_i with function *Compute_gc* according to Eq. 16 and Eq. 18. According to Eq. 17 and Eq. 19, the $(N * W)$ matrices $\boldsymbol{rc_i}$ and $\boldsymbol{rg_i}$ in D_i are initialized (lines 9-10) and *DREAM* sets each gain with the maximum value that corresponds to the case in which DCs try their best to meet the QoS demands. Finally, dividing the element one by one in $\boldsymbol{rg_i}$ by the element at the same position in $\boldsymbol{rc_i}$, we get the cost-effectiveness matrix $\boldsymbol{ce_i}$, for $i = 1, \ldots, M$.

The second phase of the algorithm is iterative, consisting of the while loop in lines 20-41. Before entering the loop, the global maximum cost-effectiveness value, ce_{max}, is computed through a collective communication operation called *all-reduce-max (send_msg, recv_msg)* [15] (line 17). The parameters are the send buffer and the receive buffer, respectively. Both are ordered lists of six variables $(ce_{max}, i, j, k, x_{ijk}, f)$, where ce_{max} is the maximum cost-effectiveness value, i, j and k are the indices of the corresponding DC, user group and channel that give rise to this maximum cost-effectiveness value. To participate in this operation each DC D_i determines its local highest cost-effectiveness value ce_{max}, the quantity of bandwidth x_{ijk} to be allocated to user group U_j from channel V_k and the flag f (lines 14 and 15). The parameter f is a flag that will be set to one if it is necessary to replicate V_k to achieve this ce_{max} in D_i, otherwise will be set to zero. Of course, when ce_{max} is greater than zero, the replica of V_k must be placed at this point, so $f = 1$.

Algorithm 1 *DREAM* ($\boldsymbol{Q}$, $\boldsymbol{V^t}$, $\boldsymbol{Y_i'}$)

1: {Datacenter D_i, $i = 1, \ldots, M$:}{Initialization}.
2: $\boldsymbol{X_i} \longleftarrow \boldsymbol{0}$; $\boldsymbol{Y_i} \longleftarrow \boldsymbol{0}$; $\boldsymbol{ce_i} \longleftarrow \boldsymbol{0}$; $f \longleftarrow 0$
3: $eS_i \longleftarrow S_i$; $eB_i \longleftarrow B_i$; $\boldsymbol{dQ} \longleftarrow \boldsymbol{Q}$; $\boldsymbol{dV^t} \longleftarrow \boldsymbol{V^t}$
4: $\boldsymbol{mc_i} \longleftarrow$ **Compute_mc**($\boldsymbol{Y_i'}$)
5: **for** $k := 1$ to W **do**
6: **if** $eS_i \geq v_k$ **then**
7: $(rg_{i0k}, rc_{i0k}) \longleftarrow$**Compute_gc**($dV_k^t, k, mc_{ik}$)
8: **for** $j := 1$ to N **do**
9: $x_{ijk} \longleftarrow min(dQ_{jk}, eB_i)$; $rg_{ijk} \longleftarrow (P_1 - C_{ij}) * x_{ijk} + rg_{i0k}$
10: $rc_{ijk} \longleftarrow C_{ij} * x_{ijk} + rc_{i0k}$; $ce_{ijk} \longleftarrow rg_{ijk}/rc_{ijk}$
11: **end for**
12: **end if**
13: **end for**
14: $ce_{max} \longleftarrow max_{jk} ce_{ijk}$; $(j, k) \longleftarrow$ arg $max_{jk} ce_{ijk}$
15: **if** $ce_{max} > 0$ **then** $f \longleftarrow 1$ **end if**
16: $send\_msg \longleftarrow (ce_{max}, i, j, k, x_{ijk}, f)$
17: **all-reduce-max**($send\_msg, recv\_msg$)
18: $(ce_{max}, i', j, k, x_{i'jk}, f) \longleftarrow recv\_msg$
19: {i' is the datacenter that has ce_{max}}
20: **while** $ce_{max} > 0$ **do**
21: **if** $i' = i$ **then**
22: {this server has the maximum cost-effective}
23: **if** $f = 1$ **then**
24: $Y_{ik} \longleftarrow 1$; $dV_k^t \longleftarrow max(0, dV_k^t - conf_i)$; $eS_i \longleftarrow eS_i - v_k$
25: **end if**
26: **if** $x_{ijk} \neq 0$ **then**
27: $X_{ijk} \longleftarrow x_{ijk}$; $dQ_{jk} \longleftarrow dQ_{jk} - x_{ijk}$; $eB_i \longleftarrow eB_i - x_{ijk}$
28: **end if**
29: **else**
30: {another data center has the maximum cost-effective}
31: **if** $f = 1$ **then** $dV_k^t \longleftarrow max(0, dV_k^t - conf_{i'})$ **end if**
32: **if** $x_{i'jk} \neq 0$ **then** $dQ_{jk} \longleftarrow dQ_{jk} - x_{i'jk}$ **end if**
33: **end if**
34: {prepare the next iteration}
35: $(\boldsymbol{ce_i}, \boldsymbol{x_i}) \longleftarrow$ **update**($\boldsymbol{dQ}, \boldsymbol{dV^t}, \boldsymbol{mc_i}$)
36: $ce_{max} \longleftarrow max_{jk} ce_{ijk}$; $(j, k) \longleftarrow$ arg $max_{jk} ce_{ijk}$
37: **if** $Y_{ik} = 0$ **then** $f \longleftarrow 1$ **else** $f \longleftarrow 0$ **end if**
38: $send\_msg \longleftarrow (ce_{max}, i, j, k, x_{ijk}, f)$
39: **all-reduce-max**($send\_msg, recv\_msg$)
40: $(ce_{max}, i', j, k, x_{i'jk}, f) \longleftarrow recv\_msg$
41: **end while**

During each iteration period, if DC D_i attains the maximum global cost-effectiveness value, it performs replica placement and bandwidth allocation if needed (lines 21-29). If D_i does not attain the maximum global cost-effectiveness value, it only updates some local values to keep track of the changes resulted from allocation of bandwidth and storage space at other DCs (lines 30-33). Then, each DC participates in another *all-reduce-max* operation that determines the next candidate maximum cost-effectiveness value ce_{max}, the quantity of bandwidth x_{ijk} and the flag f. Each DC D_i updates its cost-effectiveness matrix $\boldsymbol{ce_i}$ and $\boldsymbol{x_i}$ with the function *update* that is very similar to the codes from line 4 to line 13.

Because of possible prediction errors, the entries in demand matrix $\boldsymbol{Q}$ are not always bigger than or equal to the user groups' practical demands. So, although the constraints $\sum_{i=1}^{M} X_{ijk} \geq Q_{jk}$, $j = 1, \ldots, N$ and $k = 1, \ldots, W$, are satisfied, the reserved bandwidth may not meet the actual streaming quality with a probability. Therefore, after finishing *DREAM*, we let $X_{ijk} \longleftarrow (1 - e) * X_{ijk}$ for $i = 1, \ldots, M$, $j = 1, \ldots, N$ and $k = 1, \ldots, W$, and configure a bandwidth pool sized $e \sum_{j=1}^{N} \sum_{k=1}^{W} X_{ijk}$ in each DC, where e is a small value (e.g. 5%) and referred to as bandwidth pooling factor. A DC's pooled bandwidth can be provided to any user group for downloading any channel that is replicated in this DC. If a user group exhausts its exclusive reserved bandwidth for a specific channel, its requests will be routed to the replica with sufficient pooled bandwidth and the smallest communication distance to it.

C. Complexity

To calculate the running time of *DREAM*, we first determine the number of iterations of the main loop. We differentiate the iterations based on what they do. A *bandwidth allocating iteration* is one that only allocates bandwidth to a user group. The bandwidth allocating iteration means that no channel's replica is placed in this iteration. Similarly, a *replica placement iteration* is one that only places a channel's replica in a DC and does not allocate any bandwidth. We refer to an iteration that not only places replica but also allocates bandwidth to a user group from this replica as a *hybrid iteration*. It is clear from the algorithm description (lines 21-33) that each iteration falls into one of these three categories. Moreover, lines 5-13 show that *DREAM* does not compute the cost-effectiveness value of placing a replica alone, hence a replica placement iteration will not be carried out unless all x_{ijk}s of a pair of a DC and a channel are equal to zero.

Based on the analysis above, it is not difficult to infer that a channel may introduce at most $N - 1$ *bandwidth allocating iterations*. Thus, for all channels the maximum number of *bandwidth allocating iterations* is $W * (N - 1)$. If sufficient storage space is provided, in the worst case, a channel has to be replicated at all the DCs, which means that the total number of *hybrid and replica placement iterations* are at most $W * M$. The worst running time of each iteration is $W * N + log(W * N)$. Thus the running time of the main loop is $W * (M + N - 1) * (W * N + log(W * N))$. The initialization phase consists of a $(W * N)$-iteration loop and a *build-heap* operation of cost $(W * N)$. Therefore, the worst case running time of *DREAM* is $\boldsymbol{O((M + N) * W^2 * N)}$, where M is the total number of DCs, N is the total number of user groups, and W is the total number of channels. According to the literature [15], it is easy to prove that the communication complexity of *DREAM* is $\boldsymbol{O(W * (M + N - 1) * logM)}$.

In fact our experiments in Section IV show that each run of the *DREAM* or *DREAM-L* (Section III-D) algorithm only takes 2-4 seconds, so there is sufficient time in each 10-minute internal to determine the resource assignment scheme for the next internal.

D. Integrating Locality-Awareness

To effectively reduce the delays between users and DCs and cross-boundary traffic (e.g., cross-ISP traffic), we incorporate *DREAM* with locality-awareness, i.e. *DREAM-L*, to maximize the amount of bandwidth allocated to users from their local cloud DCs. To this end, we use an abstract notion "region" to represent the locality of a user group. In this paper, locality is defined as:

$$Locality = \sum_{j=1}^{N} aB_j / min(cB_j, rB_j), \tag{21}$$

where aB_j is the bandwidth obtained by user group U_j from the DCs located in its own region. cB_j is the sum of all DCs'

available bandwidth in this region. rB_j is the total bandwidth obtained by user group U_j.

Our objective now is to improve locality. To achieve this objective, some modifications need to be applied to the *DREAM* algorithm. First, we define virtual communication distance for transferring data units between DC D_i and user group U_j by a positive floating-point value function vc_{ij} as:

$$vc_{ij} = \begin{cases} c_{ij} & \text{if } D_i \text{ and } U_j \text{ are in the same region} \\ c_{ij} + P_3 & \text{if } D_i \text{ and } U_j \text{ are in different regions,} \end{cases} \quad (22)$$

where $i = 1, \ldots, M, j = 1, \ldots, N$ and P_3 referred to as locality penalty factors is a positive real number. Second, let $VC_{ij} = vc_{ij} * \delta t$ and replace C_{ij} with it in *DREAM*. From the definition of Eq. 19, we can see that *DREAM* tends to reserve bandwidth to users from DCs with smaller communication distance. We refer to the *DREAM* integrating locality-awareness as *DREAM-L*.

IV. PERFORMANCE EVALUATION

We conduct extensive simulations to evaluate the performance of our approach. The simulations are driven by real traces collected from the Youku site [12], which covers a 14-day duration. We first briefly introduce the collected traces, which is followed by the evaluating methodology as well as evaluation results and their analysis.

A. Traces and Parameters

YouKu is one of the largest online video and television platforms in China, reaching a total of 400 million unique viewers every month. We randomly chose 2270 videos and gathered their statistic information using two online crawlers every 10 minutes from the Youku site. The statistics of interest include video's duration, the total number of online video replays (i.e., the number of times the online video "play button" is pushed, excluding the "play" button pushed after a "pause" button to resume replay) by users and the distribution of the number of online video replays among all Chinese regions. The first crawler collected information of 1715 videos that have already been launched for a few days. The second crawler collected traces of accumulated videos launched in subsequent days at a rate of 38-40 videos per day. In our experiment the users are grouped based on their regions and there are 34 user groups (regions) totally.

The playback rate of a video is typically 500kbps, but since there are a lot of users who do not watch the entire length of the video, we follow the common practice of other researchers by assuming that each play consumes 250kbps bandwidth throughout its entire duration. We infer the size of a video by using its duration. For example, a video of 60 minutes contains $(500kbps * 60 * 60)/8/1024/1024 = 0.215GB$. We compute the incremental number of videos' online replays in every 10 minutes by the difference in the total cumulative numbers of replays between two consecutive observed periods. Using the incremental numbers of replays, videos' durations and regional distribution of total number of replays, we can form a time-series of user groups' bandwidth demands for all video channels. And then we implement time-series and demand prediction techniques [5] to forecast the expectation and variance of bandwidth demands for every 10-minute interval δt. Fig. 1 shows the forecast for users' bandwidth demands of a popular video channel from Beijing in China.

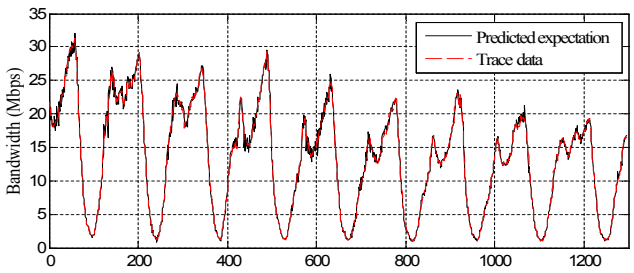

Fig. 1: 10-minute forecast conditional expectation of users bandwidth demands from Beijing in China for the video channel 142202354.

Because the cloud provider generally places servers or DCs near the users, in our experiments, 128 DCs are assumed to be uniformly distributed in 34 regions. We conduct three experiments with each DC's available bandwidth capacity assumed to be 1.8Gbps, 10Gbps and 25Gbps respectively. The assumed cloud bandwidth capacity in the first experiment (i.e., 1.8Gbps per DC) is closer to the actual total bandwidth demands of the users. Each DC's bandwidth capacity in the second (i.e., 10Gbps) and the third (i.e., 25Gbps) experiments are assumed based on the current 10Gpong and next generation 25Gpong communication components in the cloud. We assume that all DCs have the same confidence level and each video needs 2 or 3 replicas to maintain its availability. We assume the three penalty factors to be $P_1 = 20$, $P_2 = 1$, and $P_3 = 1/6$. $Pr(L_{jk} > Q_{jk}) \leq \varepsilon$ and $e \sum_{j=1}^{N} \sum_{k=1}^{W} X_{ijk}$ are set to $\varepsilon = 40\%$ and $e = 5\%$, for $i = 1, \ldots, M$, $j = 1, \ldots, N$ and $k = 1, \ldots, W$, where M, N and W are the numbers of DCs, user groups and channels respectively. We set the storage cost factor and communication distance with storage and data transfer pricing similar to those in Amazon's simple storage service (S3) of July 2013.

B. Methodology

We compare our *DREAM* and *DREAM-L* (*DREAM* with locality-awareness) with the *Per-DC Lim* and *Optimal Load Direction* [5], which are considered the state-of-the-art channel-based solutions and referred to as *Niu-1* and *Niu-2* in this paper respectively. As pointed out by Niu et al. [5], the *Niu-2* algorithm stores a large number of video replicas, which incurs significant additional cloud-charged storage fees to the VoD provider. Therefore, they overcame this shortcoming by limiting the number (k) of channels stored in each DC in the *Niu-1* algorithm. We set $k = 5$, when there are 128 DCs available.

We use the following ten metrics in our simulation, three for measuring quality of service, three for system configuration, and four for measuring costs. The three QoS metrics are *Service availability*, the arithmetic mean of all channels' availability during a 10-minute interval, *Service access locality*, the arithmetic mean of all user groups' locality, and *Streaming quality*, the ratio of bandwidth required by users to their actual bandwidth demand. The three system measures are *Replication degree*, the average number of replicas per video channel, *Over-provisioning ratio*, the ratio of total reserved

TABLE II: PERFORMANCE OF THE 4 ALGORITHMS WITH DIFFERENT DATACENTER BANDWIDTH CAPACITIES DURING 10 DAYS

(a) Each datacenter has 1.8Gbps bandwidth.

Algorithm	QoS Measures(averaged over all intervals)			System Measures(averaged over all intervals)				Cost Measures(sum over all intervals)		
	Avail	*Stq*	*Local*	*Rep*	*Opr*	*Crt*	*Dc($)*	*Sc($)*	*Mc($)*	*Oc($)(normalized to Niu-1)*
DREAM	100%	99.94%	2.97%	2.62	102.80%	101.01%	353619	6.76	412.14	354038 (60.39%)
DREAM-L	100%	99.94%	96.89%	27.94	102.86%	10.65%	458940	76.61	1374.08	460391 (78.53%)
Niu-1	35.14%	99.96%	2.94%	2.12	105.36%	100%	557467	5.81	28851.54	586260 (100%)
Niu-2	100%	99.97%	3.00%	33.11	103.35%	100.21%	554160	88.58	7526.45	561775 (95.82%)

(b) Each datacenter has 10Gbps bandwidth.

Algorithm	QoS Measures(averaged over all intervals)			System Measures(averaged over all intervals)				Cost Measures(sum over all intervals)		
	Avail	*Stq*	*Local*	*Rep*	*Opr*	*Crt*	*Dc($)*	*Sc($)*	*Mc($)*	*Oc($)(normalized to Niu-1)*
DREAM	100%	99.94%	2.95%	2.59	102.85%	101.89%	256434	6.53	142.81	256583 (49.67%)
DREAM-L	100%	99.94%	96.45%	28.75	102.86%	0.35%	394166	77.81	285.21	394529(76.38%)
Niu-1	13.23%	99.96%	2.83%	2.00	105.32%	100%	491709	5.20	26346.73	516549(100%)
Niu-2	86.12%	99.97%	2.94%	6.39	103.35%	101.22%	489353	17.27	1365.28	490736 (95.00%)

(c) Each datacenter has 25Gbps bandwidth.

Algorithm	QoS Measures(averaged over all intervals)			System Measures(averaged over all intervals)				Cost Measures(sum over all intervals)		
	Avail	*Stq*	*Local*	*Rep*	*Opr*	*Crt*	*Dc($)*	*Sc($)*	*Mc($)*	*Oc($)(normalized to Niu-1)*
DREAM	100%	99.94%	3.17%	2.57	102.86%	101.59%	244644	6.31	34.80	244685 (50.31%)
DREAM-L	100%	99.94%	96.47%	28.75	102.86%	0.34%	383253	77.87	190.08	383521 (78.86%)
Niu-1	13.88%	99.96%	2.81%	2.13	105.25%	100%	461161	5.55	28272.34	486354 (100%)
Niu-2	64.53%	99.97%	2.94%	2.86	103.35%	100.66%	421173	8.03	485.60	421667 (86.70%)

Avail: Service availability; *Stq*: Streaming quality; *Local*: Service access locality; *Rep*: Replication degree; *Opr*: Over-provisioning ratio; *Crt*: Normalized cross-region traffics; *Dc*: Delivery costs; *Sc*: Storage costs; *Mc*: Migration costs; *Oc=Dc+Sc+Mc*: Operating costs(Normalized operating costs). Cross-region traffic and operating cost measure are normalized to those of the *Niu-1* algorithm.

bandwidth to users' actual bandwidth demand, and *Cross-region traffic*, the amount of traffic that crosses different regions. The four cost metrics, *Delivery cost*, *Storage cost*, *Migration cost* and *Operating cost*, have been defined formally in Section II. Informally, *Delivery cost* is the communication cost of delivering data from the cloud to users and *Migration cost* is the communication cost of moving data among DCs; *Storage cost* is the overhead of storing data in the cloud. In this paper, *Operating cost* is the sum of *Delivery cost*, *Storage cost* and *Migration cost*. Note that all costs, in US dollars, are derived based on the similar pricing scale of Amazons S3 as of July 2013.

C. Performance Observed at VoD Service Providers

We implement the following four resource scheduling algorithms for performance comparison, *DREAM*, *DREAM-L*, *Niu-1* and *Niu-2*. Table II shows the three QoS measures, three system measures, and four cost measures of different algorithms during a 10-day period. Our *DREAM* and *DREAM-L* have lower operating costs, respectively achieving 49.67%-60.39% and 76.38%-78.86% of the cost of the *Niu-1* algorithm, and 52.29%-63.02% and 80.40%-90.95 of the cost of the *Niu-2* algorithm. In other words, our algorithms can save 9.05%-50.33% total cloud operating costs of *Niu-1* and *Niu-2*.

Because *Niu-1* and *Niu-2* do not consider the optimization of the delivery cost, they incur higher delivery costs than *DREAM* and *DREAM-L*. Our algorithms obtain 52%-91% of their delivery costs. Following the most common practices of pricing in the cloud platform, in our experiments the communication price is assumed to be independent of user's location. However, since DCs in different regions in the cloud platform are known to charge differently for the same bandwidth capacity, some users may be assigned more expensive bandwidth from closer DCs. The locality awareness of *DREAM-L* means that sometimes users may end up obtaining more expensive bandwidth from local DCs, which explains why the delivery cost of *DREAM-L* is higher than that of *DREAM*. Meanwhile, to provide access locality, *DREAM-L* must move and store more channel replicas to the regions where users' requests are originated, so *DREAM-L* has higher storage and migration costs than *DREAM*. Comparing *DREAM-L* with *DREAM*, the access locality provided by the former incurs 30%-57% more operating cost than the latter. It may be argued that, for better access locality, in some situations it is necessary to pay for this additional cost. In general, such a performance-vs.-cost tradeoff can often be made at the SLA negotiation time. When compared with *Niu-1* and *Niu-2*, however, *DREAM-L* offers consistent and impressive operating-cost advantages, as shown in Table II.

The *Niu-1* algorithm limits the number of channels stored in each DC, so it has lower storage cost, achieving 6.57%-69.11% of that of the *Niu-2* algorithm. But this limitation on the number of channels per DC severely weakens the service availability of *Niu-1*, as shown in Table II.

Both *Niu-2* and *Niu-1* reset the layout of channels in every interval, while our approaches try to reduce the operating cost including channel migration cost. So, our algorithms' migration costs are only small fractions of those of *Niu-1* and *Niu-2*. In addition, because many replicas of channels have already been stored in the DCs, *Niu-2* is able to avoid some migration operations of *Niu-1*. On the other hand, when the DCs have larger bandwidth capacities, *Niu-2*, by reserving bandwidth from and placing replicas in a very small subset of DCs, is able reduce storage cost and migration cost further. In addition, as shown in Table II, our two algorithms book the minimum necessary bandwidth and achieve comparable streaming quality to *Niu-1* and *Niu-2*.

In our experiments, we found that the storage costs of the four algorithms are so small that they can be practically neglected. This means that the network communication cost including delivery and migration cost is the dominant ele-

ment of VoD applications operating cost in the current cloud platform, accounting for at least 99% operating costs of the four approaches. Therefore, we can conclude that the network bandwidth is still an expensive resource and it is of paramount importance to develop bandwidth optimization strategies for VoD applications.

D. QoS of VoD Service

As shown in Table II, because of integrating availability constraint (Constraint 11) in our model, our algorithms can meet all channels' availability demands completely. Since *DREAM-L* must replicate channels into many regions to allow users to access data from local DCs, it has a high replication degree. In Table II, of all the algorithms that meet all availability demands, *DREAM* has the lowest storage cost and replication degree (replicas per channel). *Niu-1* and *Niu-2* do not provide any strategy for maintaining availability. But when the DCs have smaller bandwidth capacities, *Niu-2* must place replicas into many more DCs in order to reserve and obtain sufficient bandwidth from them. Therefore, it can meet all availability demands under this condition. But with the expansion of DCs' bandwidth capacities, the number of replicas is reduced by *Niu-2*, hence the service availability drops to a lower level as shown in Table II(b) and Table II(c). In all cases, *Niu-1* provides the lowest availability. As shown in Table II, all four algorithms provide comparable streaming quality, and have similar over-provisioning ratio.

Now we investigate the access locality. As shown in Table II, *DREAM-L* significantly improves the access locality by integrating locality awareness. This means that *DREAM-L* utilizes the vast majority of the local bandwidth resources to serve the users so as to reduce the startup latency and cross-region traffic. The *Cross-region traffic* metric not only shows the locality awareness of an algorithm but also is a general performance indicator as lower cross-region traffic means that users are closer to their servers. Table II shows the cross-region traffic generated by different algorithms, normalized to that of the *Niu-1* algorithm. As the *DREAM*, *Niu-1* and *Niu-2* algorithms do not have locality awareness, it is thus not surprising to see their cross-region traffic much higher than that of the *DREAM-L* algorithm.

V. Conclusion

In this paper, we propose a holistic approach to addressing the fundamental resource reservation and scheduling challenge of how to configure the cloud utility to meet SLAs for VoD applications at a modest cost. We first devise an optimization model that describes the relationship among channel placement, bandwidth allocation, cloud costs and QoS constraints including streaming quality and data availability. Using this model, the researchers and developers can deeply view how the VoD works in cloud platform from the cost elements, resources assignment and QoS demands points. And then we propose distributed heuristic algorithms *DREAM*and *DREAM-L* that integrate *DREAM* with locality-awareness to solve the model. Leveraging demand prediction and executed at a 10-minute frequency, the algorithms dynamically determine the video layout and reserve cloud bandwidth from multiple DCs to meet SLA and satisfy the QoS in a cost-effective way. From extensive simulations driven by the demand traces of a large scale real-world VoD system, we observe that, compared to the existing state-of-the-art algorithms, our algorithms are able to provide perfect data availability and high access locality, and achieve comparable streaming quality at only 50%-90% of the cloud cost of the existing state-of-the-art algorithms.

Acknowledgment

This research is partially supported by the National Basic Research Program (973 Program) of China under Grant No. 2011CB302305, and the National Natural Science Foundation of China under Grant No. 61232004. This research is also supported by Wuhan National Laboratory for Optoelectronics and Key Laboratory of Information Storage System, Ministry of Education. The authors thank graduate student Ji Chen and Dr. Zhaolu Guo for providing crawlers to help us collect traces.

References

[1] H. Li, L. Zhong, J. Liu, B. Li, K. Xu, "Cost-Effective Partial Migration of VoD Services to Content Clouds," in *Proc. IEEE International Conference on Cloud Computing (CLOUD)*, 2011.

[2] "Four Reasons We Choose Amazon's Cloud as Our Computing Platform," The Netflix "Tech" Blog, 2010.

[3] X. Hei, C. Liang, J. Liang, Y. Liu, K. W. Ross, "A Measurement Study of a Large-Scale P2P IPTV System," *IEEE Transactions on Multimedia*, vol.9, no.8, pp.1672–1687, 2007.

[4] J. Liu, S. G. Rao, B. Li, H. Zhang, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast," *Proceedings of the IEEE*, vol.96, no.1, pp.11–24, 2008.

[5] D. Niu, H. Xu, B. Li, S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proc. IEEE INFOCOM*, 2012.

[6] Y. Wu, C. Wu; Bo Li, X. Qiu, F. C. Lau, "CloudMedia: When Cloud on Demand Meets Video on Demand," in *Proc. 31st International Conference on Distributed Computing Systems (ICDCS)*, 2011.

[7] N. Carlsson, G. Dan, D. Eager, A. Mahanti, "Tradeoffs in cloud and peer-assisted content delivery systems," in *Proc. IEEE 12th International Conference on Peer-to-Peer Computing (P2P)*, 2012.

[8] F. Wang, J. Liu, M. Chen, "CALMS: Cloud-assisted live media streaming for globalized demands with time/region diversities," in *Proc. IEEE INFOCOM*, 2012.

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

[10] Windows Azure, http://www.windowsazure.com/.

[11] C. Chekuri, S. Khanna, "A polynomial time approximation scheme for the multiple knapsack problem," *SIAM Journal on Computing*, vol.35, no.3, pp.713–728, 2005.

[12] Youku, http://www.youku.com/.

[13] A. Mahimkar, A. Chiu, R. Doverspike, M. D. Feuer, P. Magill, E. Mavrogiorgis et al., "Bandwidth on Demand for Inter-Data Center Communication," in *Proc. the 10th ACM Workshop on Hot Topics in Networks (Hotnets)*, 2011.

[14] T. Loukopoulos, I. Ahmad, D. Papadias, "An overview of data replication on the Internet," in *Proc. the 6th International Symposium on Parallel Architectures Algorithm and Networks*, 2002.

[15] A. Grama, G. Karypis, V. Kumar, and A. Gupta, "Introduction to Parallel Computing," second ed., ch.4, Addison Wesley, 2003.

[16] Z. Lu, J. Wu, Y. Huang; L. Chen, D. Deng, "CPDID: A Novel CDN-P2P Dynamic Interactive Delivery Scheme for Live Streaming," in *Proc. IEEE 18th International Conference on Parallel and Distributed Systems (ICPADS)*, 2012.

[17] N. Bonvin, T. G. Papaioannou, and K. Aberer, "A selforganized, fault-tolerant and scalable replication scheme for cloud storage," in *Proc. 1st ACM symposium on Cloud computing*, 2010.

[18] S. U. Khan, I. Ahmad, "Comparison and analysis of ten static heuristics-based Internet data replication techniques," *Journal of Parallel and Distributed Computing*, vol.68, no.2, pp.113–136, 2008.