DREAM-(L)G: A Distributed Grouping-Based Algorithm for Resource Assignment for Bandwidth-Intensive Applications in the Cloud

Yuhong Zhao, Hong Jiang, *Fellow, IEEE*, Ke Zhou, *Member, IEEE*, Zhijie Huang, and Ping Huang, *Member, IEEE*

Abstract—Increasingly, many bandwidth-intensive applications have been ported to the cloud platform. In practice, however, some disadvantages including equipment failures, bandwidth overload and long-distance transmission often damage the QoS about data availability, bandwidth provision and access locality respectively. While some recent solutions have been proposed to cope with one or two of disadvantages, but not all. Moreover, as the number of data objects scales, most of the current offline algorithms solving a constraint optimization problem suffer from low computational efficiency. To overcome these problems, in this paper we propose an approach that aims to make fully efficient use of the cloud resources to enable bandwidth-intensive applications to achieve the desirable level of SLA-specified QoS mentioned above cost-effectively and timely. First we devise a constraint-based model that describes the relationship among data object placement, user cells bandwidth allocation, operating costs and QoS constraints. Second, we use the distributed heuristic algorithm, called *DREAM-L*, that solves the model and produces a budget solution to meet SLA-specified QoS. Third, we propose an object-grouping technique that is integrated into *DREAM-L*, called *DREAM-LG*, to significantly improve the computational efficiency of our algorithm. The results of hundreds of thousands of simulation-based experiments demonstrate that *DREAM-LG* provides much better data availability, bandwidth provision and access locality than the state-of-the-art solutions at modest cloud operating costs and within a small and acceptable range of time.

Index Terms—Bandwidth-intensive applications, cloud system, quality of service, distributed heuristic algorithm, resource scheduling, optimization model

1 Introduction

Bandwidth-intensive applications, such as sharing files and software [9], watching online videos [10], and listening to online music, have become one kind of the most popular services on the Internet. Generally speaking, bandwidth-intensive applications share some common characteristics, including consuming enormous amounts of bandwidth and having a large number of data objects (an example of a data object can be a file, a video channel, a data chunk or block, etc.) and geo-users. In general, bandwidth-intensive applications deployments use either CDNs (Content Delivery Networks) [37] or P2P (Peer-to-Peer) networks [36]. While the former achieve high availability, the latter have lower deployment cost and are more scalable.

With the prevalence of the cloud computing and storage platforms, service providers for the bandwidth-intensive applications increasingly rely on these cloud platforms to support their applications that in turn provide content and service to global users [20], [21], [22]. In the past decade, the cloud architecture has evolved from a single high-bandwidth data-center (DC) towards a much larger delivery infrastructure composed of multiple data-centers (DCs) distributed over vast areas [5], [6]. Such geo-distributed clouds provide a more effective and economic solution, that is, the on-demand resource provision in the cloud aims to meet the dynamic bandwidth and storage demands of the bandwidth-intensive applications in real time [11] and cloud DCs located in different geographic locations are able to provide efficient services to cells of users in their proximity.

However, downtime caused by DCs' failures and overload can incur significant economic damages to application providers and users. In 2010, for example, North American businesses lost an estimated \$26.5 billion in revenue due to partial or complete outage of services [17]. Even worse, a 2013 survey [1] reports that the unplanned outages cost more than \$7,900 per minute, in increase of 40.8 percent from 2010(\$5,600). In practice, failures of storage, network breakdowns and other causes [2], [3], [14], [15], [16], [41] can cause portions of or even an entire DC with thousands to millions of data replicas to become unavailable. At the same time, increase in workload and network traffic can also overload the bandwidth both in and out of some cloud DCs [23], causing spikes in network latencies. In both cases, Internet applications become unavailable and/or unresponsive. To make this problem even worse, many modern-day

- *Y. Zhao, K. Zhou, and Z. Huang are with the School of Computer Science and Technology, Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System, Ministry of Education, Huazhong University of Science and Technology, Wuhan, China. E-mail: zhaoyuhong1945@163.com, {k.zhou, jayzy_huang}@hust.edu.cn.*
- *H. Jiang is with the University of Texas at Arlington, TX. E-mail: jiang@cse.unl.edu.*
- *P. Huang is with the Virginia Commonwealth University, Virginia, VA. E-mail: phuang@vcu.edu.*

Manuscript received 21 Apr. 2015; revised 2 Feb. 2016; accepted 14 Feb. 2016. Date of publication 2 Mar. 2016; date of current version 16 Nov. 2016. Recommended for acceptance by Z. Lan. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TPDS.2016.2537334

applications have become highly globalized, with users coming from all over the world [8] making it more likely for bandwidth-intensive applications to become unavailable or unresponsive due to the sometimes extremely long transmission distances.

In this paper, we present practical solutions to some of the above problems that account for the requirements of both the underlying bandwidth-intensive applications and the global users. Given a general cloud platform consisting of multiple DCs, our goal is to cost-effectively improve the availability of bandwidth-intensive applications with multiple data objects, provide sufficient DCs' bandwidth to users as well as access locality. However, fulfilling all the requirements may necessitate very large potential investment into the cloud resources, which application providers would certainly like to avoid or minimize. Therefore, our other goal is to minimize the cloud costs that the application providers have to invest.

However, it remains a challenge to simultaneously improve the application availability, bandwidth provision and access locality while reducing the costs of cloud applications. More specifically, this challenge can be explained from the following perspectives.

- While it is possible to improve the application availability by spreading data replicas of a particular application across many DCs to reduce the impact of any single DC's failure, this allocation strategy comes at a higher cost in storage capacity and communication bandwidth resulting from storing and updating replicas.
- Although the total bandwidth purchased from the cloud platform is more than the sum of all users' bandwidth demands, individual DCs' limited bandwidth capacity and possible overload of some DCs can easily render an application unavailable for some users.
- To improve access locality, one may consider placing the replicas of data objects into and allocating bandwidth from the DCs that are near the users who want to access these data objects, which complicates replica placement and bandwidth allocation.
- Globalization of DCs and users makes user behaviors and demands even more dynamic [8]. The majority of current algorithms solving an optimization problem scale poorly, suffering from low computational efficiency [13]. The running time of these optimization algorithms can become as long as thousands of seconds, which makes it unacceptable for the applications that need to adjust their resource assignment schemes timely to meet users' dynamic demands.

Table 1 lists three most relevant optimization algorithms [3], [4] designed to optimize one or two of the three key QoS requirements of application availability, sufficient bandwidth for users and access locality. To understand these algorithms, we applied them to the simulation experiments and assessed their performance, in terms of application availability, the bandwidth provision and access locality. As expected, all these algorithms significantly improve the metric(s) they were designed to optimize for, but unfortunately ignore or degrade the performance in the unoptimized metrics. Thus, *our goal in this paper is to design algorithms capable of producing data replica placement and bandwidth allocation that cost-effectively achieve optimization on all the three QoS requirements, i.e., application availability, bandwidth provision and access locality simultaneously for bandwidth-intensive applications in the cloud*. Like the algorithms in Table 1, our algorithms operate at the granularity of a datacenter.

TABLE 1
The Existing, Most Relevant Optimization Algorithms for Application Availability, Bandwidth Provision and Access Locality, Where "**" Means Very Effective, "*" Means Moderately Effective, and the Blank Means Unoptimized or Poor

Algorithms	Availability	Bandwidth	Locality
Decision Tree	**		*
Per-DC Lim		**	
Optimal Load	*	**	

Our work is motivated by the following observations and insight:

- In a cloud platform, different DCs have different prices for storage and data transfer. Thus, judicious purchases of cheaper cloud resources can be very helpful in reducing the investment of application providers.
- It is important and necessary to schedule DCs' bandwidth to avoid bandwidth overload and provide sufficient bandwidth to users.
- To meet all QoS requirements mentioned above cost-effectively, it is very necessary to understand the inherent relationship among QoS requirements, cloud resource assignment and application operating costs consisting of storage cost and data transfer cost, and to establish an optimization problem that transforms the relationship to a mathematic model.
- When the scale of an optimization problem reaches a certain level, it may take too a long time to generate its resolution in acceptable amount of time even for a heuristic algorithm. Thus, it is important to speedup the algorithm by parallelization.

With this in mind, the main ideas behind and thus contributions of our proposed approach are threefold.

1) We devise a constraint-based optimization model that describes the relationship among object replica placement, DCs' bandwidth allocation, total operating costs, and QoS constraints.
2) Aiming to satisfy the QoS requirements of object availability and bandwidth provision in a cost-efficient way, we use our *d*istributed heuristic *re*source scheduling *a*lgorith*m*, called *DREAM* and presented in our previous work [38], which produces a budget solution that determines the replica layout among DCs, and tries its best effort to reserve and allocate cloud bandwidth. This algorithm is then extended to *DREAM-L* that incorporates *DREAM* with locality awareness by adjusting a coefficient.
3) We propose and incorporate a data object grouping technique into *DREAM*(*-L*), called *DREAM-G*(*-LG*), which reduces the running time of our optimization algorithm to an acceptable range but at a small and acceptable cost of scheduling quality. By clustering

the objects into many small object groups, this grouping technique allows the optimization algorithm to be executed in all groups in parallel, substantially cutting down the running time.

Our extensive evaluation of *DREAM-G*, driven by real-world traces and detailed in Sections 5 and 6, demonstrates that our approach significantly outperforms three state-of-the-art algorithms in simultaneously meeting the aforementioned QoS requirements cost-effectively within a small range of time. These state-of-the-art algorithms are the general solutions *Decision Tree (DT)* algorithm [3], and two VoD-specific solutions, the *Optimal Load Direction* algorithm and the *Per-DC Limited Channels* algorithm [14]. In addition, we demonstrate in Section 5 that by applying a load-prediction scheme [14], our offline optimization algorithm is able to achieve the same real-time responsiveness as the online *Decision Tree* algorithm [3].

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 first introduces the system architecture, and then presents our mathematical model for resource scheduling. We develop and analyze the heuristic optimization algorithm *DREAM-(L)G* used to solve the model in Section 4. Two sets of empirical evaluations of *DREAM-(L)G* algorithm are presented in Sections 5 and 6 to compare their performances to those of the state-of-the-art solutions of *Decision Tree*, *Optimal Load Direction*, and *Per-DC Limited Channels*. And Section 7 concludes our paper.

2 Related Work

The bandwidth scheduling and replica placement problem we are considering has some similarities to several other optimization problems, such as the multiple knapsack problem [26], [28], the facility location problem [29], the generalized assignment problem [27], and the transportation problem [30], [31]. Previous research on replica placement [3], [39], [40] and bandwidth allocation mechanisms [4], [7] has largely focused on improving application availability by providing multiple replicas among servers or DCs and on ensuring predictable network behavior through better bandwidth allocation and reservation.

The work closest to ours is by Bonvin et al. [3], which proposed an online self-organized replication scheme that tries to dynamically maintain availability guarantee to data objects, place the replicas close to users to offer access locality and balance the user-request load among all servers. The scheme allows each replica of a data object in cloud storage platform to act as an individual optimizer that periodically executes a so-called *Decision Tree* algorithm to choose whether to migrate, replicate or remove itself based on the net benefit maximization in terms of the utility offered by the item and its storage and maintenance cost.

Focusing on VoD applications in the cloud environment, Niu et al. [4] proposed a customized predictive resource auto-scaling system that dynamically places replicas of channels and books the minimum bandwidth resources from multiple DCs to match the users' next short-term bandwidth demand with high probability. First, they predict the conditional mean of users' bandwidth demand. Second, based on the prediction and the assumption that users' total bandwidth demand for each video channel in a short period follows a Gaussian distribution, their algorithms are

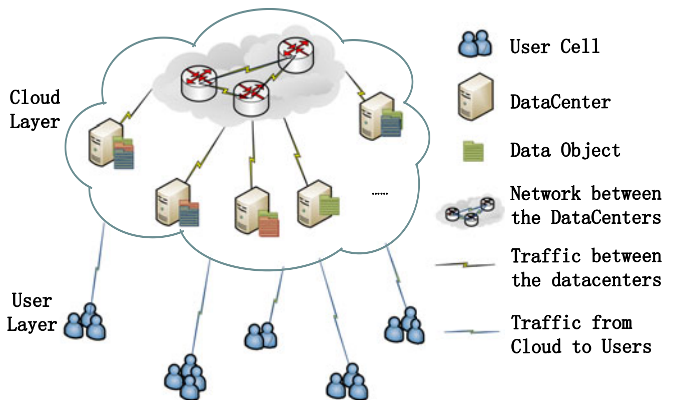

Cloud Layer
User Layer
User Cell
DataCenter
Data Object
Network between the DataCenters
Traffic between the datacenters
Traffic from Cloud to Users

Fig. 1. An overview of the system architecture.

designed to reserve and assign bandwidth from multiple DCs to each channel to obtain desirable streaming quality.

Focusing on the data replica placement problem in distributed systems, Ahmad et al. [32], [33] presented a unified cost model that captures the minimization of the total data object transfer cost, which in turn leads to effective utilization of storage space, and replica consistency. Then they compared and analyzed ten heuristics to solve this model [13]. Aiming to provide cost-effective availability, and improve performance and load balance of cloud storage, Wei et al. [39] presented a cost-effective dynamic replication management scheme. By exploiting the skewness in the observed communication patterns, Bodik et al. [2] proposed a novel optimization framework that achieves both high fault tolerance and significantly reduces bandwidth usage in the network core. Castillo et al. [34] proposed a replication technique, which takes into account resource usage, system reliability and the characteristic of the dataflow to decide what data to replicate and when to replicate. To improve the efficiency of object replication, Zaman and Grosu [12] proposed a distributed algorithm *DGR* that replicates data objects to a group of servers with different cache sizes to minimize the total time taken by all servers accessing all data objects.

3 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 data object placement, QoS constraints, bandwidth assignment and cloud operating costs for bandwidth intensive applications in the cloud. The system architecture under consideration, as shown in Fig. 1, is a large-scale distributed system, where users access data objects that are held by the datacenters.

3.1 System Architecture and Notation Definition

The most frequently used notations are listed in Table 2. Without loss of generality, we assume the cloud platform to be composed of M datacenters. Let D_i, S_i, dB_i and uB_i be the name, the total storage capacity and bandwidth capacity of delivering and receiving data, respectively, of DC i where $i = 1, \dots, M$. These DCs are geographically distributed among different regions, and they can communicate with one another through a network (e.g., WAN).

TABLE 2
Notations and Their Meanings

Notations	Meaning
M	Total number of datacenters in the system
N	Total number of user cells
W	Total number of data objects 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
dB_i	DC D_i's bandwidth capacity of delivering data
uB_i	DC D_i's bandwidth capacity of receiving data
U_j	The jth user cell
oc_{ij}	Communication distance from DC D_i to user cell U_j
ic_{ij}	Communication distance from user cell U_j to DC D_i
V_k	The kth data object
v_k	Size of the data object V_k
V_k^{th}	The threshold of data object 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
dQ_{jk}	The expected bandwidth of U_j when it downloads object V_k
uQ_{jk}	The expected bandwidth of U_j when it uploads data of object V_k
X_{ijk}	The bandwidth assigned by DC D_i to user cell U_j for downloading object V_k
Z_{ijk}	The bandwidth assigned by DC D_i to user cell U_j for uploading or updating object V_k
Y_{ik}	If V_k is replicated at D_i, $Y_{ik} = 1$, otherwise $Y_{ik} = 0$
$avail(V_k)$	The availability of the object V_k
δt	The interval between two runs of the algorithm
$C_{\delta t}$	The operational cost of an application in δt

We assume that there are W data objects (a data object can be a file, a video channel, a data chunk or block, etc.). Each object 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.

To expose and exploit access locality, we differentiate users into cells based on their network status. A user cell 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)). The mapping of users into cells, which is out of scope of this paper, is assumed to be the responsibility of the application provider or developer. There are N user cells, $U_1, \ldots, U_N$, downloading, uploading and updating data objects from the cloud platform. They have different demands for upload and download bandwidth. Users can access the cloud platform through the Internet. Each route between data center D_i, $i = 1, \ldots, M$ and user cell U_j, $j = 1, \ldots, N$ has two positive floating-point value functions oc_{ij} and ic_{ij}, specifying the communication distance for transferring data units from D_i to U_j and from U_j to D_i respectively. The communication distance, determined by the locations of the DC and, sometimes, the user cell, can be represented in terms of the data transfer price between two regions [5], [6], or cost of leasing the network links. Traditionally, a cloud provider leases or owns dedicated lines connecting its DCs [18], 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 cell.

3.2 QoS Metrics of Interest

In this paper we consider the following three QoS metrics as objectives of optimization.

Data availability. We adopt the definition of availability proposed in [3] to approximate the potential availability of a data object. The availability of a data object V_k is defined as the sum of diversity of each distinct pair of DCs hosting V_k's replicas, i.e.:

$$\begin{aligned} avail(V_k) = \sum_{(i=1:Y_{ik}=1)}^{M} \sum_{(l=i+1:Y_{lk}=1)}^{M} \\ conf_i * conf_l * diversity(D_i, D_l) \quad \geqslant V_k^{th}, \end{aligned} \tag{1}$$

where $conf \in [0, 1]$ 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. $conf_i$ and $conf_l$ are the confidence level of DC D_i and D_l respectively. Data availability generally increases with the geographical diversity of the selected servers. So the $diversity$ function is introduced in [3], which returns a number calculated based on the geographical distance between each DC pair. Let $Y_{ik} = 1$ if DC D_i holds a replica of object V_k during an interval δt, and 0 otherwise. The cloud platform always aims to keep the data availability above a minimum level. We assume that all data objects have diverse availability requirements, so we set different availability thresholds for them. Object V_k's availability threshold is denoted by V_k^{th}.

Access locality. We use an abstract notion "region" to represent the locality of a user cell. In this paper, a user cell's download locality is defined as:

$$Download\ Locality(U_j) = aB_j / \min(cB_j, dB_j), \tag{2}$$

where aB_j is the download bandwidth obtained by user cell U_j from the DCs located in its own region, cB_j is the sum of all DCs' bandwidth capacity of delivering data within this region, and dB_j is the total download bandwidth obtained by user cell U_j.

Similarly, we can obtain the definition for upload locality by replacing corresponding variables in Eq. (2). And we define a user cell's access locality as the arithmetic mean of download and upload locality, that is:

$$\begin{aligned} Access\ Locality(U_j) = (Download\ Locality(U_j) \\ + Upload\ Locality(U_j))/2. \end{aligned} \tag{3}$$

Bandwidth provision. dQ_{jk} and uQ_{jk} the download and upload bandwidth demands of user cell U_j for accessing object V_k respectively during an interval δt. This means that U_j should acquire sufficient bandwidth, no less than dQ_{jk} to download V_k, and no less than uQ_{jk} to update V_k, otherwise the users in U_j will experience poor QoS. The values of dQ_{jk} and uQ_{jk} ($j = 1, \ldots, N$ and $k = 1, \ldots, W$) are derived from profiling [13] or prediction in advance [4]. A user cell's bandwidth provision is defined as:

$$\begin{aligned} &Bandwidth\ Provision(U_j)\\ &= (1/W) * \sum_{k=1}^{W} \{min(dB_{jk}/dQ_{jk}, 100\%)\\ &\quad + min(uB_{jk}/uQ_{jk}, 100\%)\}/2, \end{aligned} \tag{4}$$

where dB_{jk} and uB_{jk} are the download and upload bandwidth provided to user cell U_j for accessing object V_k.

3.3 Optimization Model

All the bandwidth demands dQ_{jk} and uQ_{jk} ($j = 1, \ldots, N$ and $k = 1, \ldots, W$), which are distributed to all DCs, define two $N * W$ demand matrixes ***dQ*** and ***uQ*** respectively. Now, we use $\boldsymbol{X}$ and $\boldsymbol{Z}$ to denote the reserved download and uploading bandwidth distribution matrix as follows. Each element of the $M * (N * W)$ matrixes $\boldsymbol{X}$, X_{ijk}, is a real number representing the bandwidth provided by DC D_i for user cell U_j to download object V_k during the interval δt. Similarly, each element of the $M * (N * W)$ matrixes $\boldsymbol{Z}$, Z_{ijk}, is a real number representing the bandwidth provided by DC D_i for user cell U_j to upload or update object V_k. To maintain the bandwidth provision of the interval, we must ensure that $\sum_{i=1}^{M} X_{ijk} \geq dQ_{jk}$ and $\sum_{i=1}^{M} Z_{ijk} \geq uQ_{jk}$, $j = 1, \ldots, N$ and $k = 1, \ldots, W$. Let $Y_{ik} = 1$ if D_i holds a replica of object V_k during interval δt, and 0 otherwise. Y_{ik}, $i = 1, \ldots, M$ and $k = 1, \ldots, W$, define an $M * W$ replication matrix $\boldsymbol{Y}$ with Boolean elements. One of our model's goals is to minimize the total operating cost, denoted as $C_{\delta t}$, which will be incurred by bandwidth-intensive applications during the short-term δt. Since the communication cost of the control messages has minor impact to the overall performance of the system [13], [35], we do not consider it in our model. Now, we formally define all four components that constitute the operating cost. The first component of the operating cost, *SC*, is the storage cost arising from storing application providers' data, defined as follows:

$$Sc = \sum_{k=1}^{W} \sum_{i=1:Y_{ik}=1}^{M} Sc_{ik}, \tag{5}$$

$$Sc_{ik} = k_i * v_k * \delta t. \tag{6}$$

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. So, Sc_{ik} is the storage cost of object V_k sized v_k in D_i during δt.

The second component of the operating cost, *Mc*, is the replica migration cost due to the inter-DC movement of object replicas as a result of the optimization algorithm's decision on replicas' new locations:

$$Mc = \sum_{k=1}^{W} \sum_{i=1:Y_{ik}=1}^{M} Mc_{ik}, \tag{7}$$

$$Mc_{ik} = (dc_i^k * v_k) * (1 - Y'_{ik}). \tag{8}$$

Generally the cost of replicating an object from one DC to another is proportional to the object's size. We refer to this cost as migration cost and use $dc_i^k * v_k$ to calculate it, where dc_i^k is the communication distance from D_i to its "nearest" neighboring DC that has object V_k's replica. We set $Y'_{ik} = 1$ when the object V_k sized v_k is already replicated in DC D_i by the last execution of the optimization algorithm, otherwise it is equal to 0. Obviously, Eq. (8) will be zero if V_k's replica was already placed in D_i by the last execution of the algorithm (i.e., $Y'_{ik} = 1$).

The third component, *Dc*, is the delivery cost that is introduced by sending data from DCs to users:

$$Dc = \sum_{k=1}^{W} \sum_{i=1}^{M} \sum_{j=1}^{N} Dc_{ijk}(X_{ijk}), \tag{9}$$

$$Dc_{ijk}(X_{ijk}) = oc_{ij} * X_{ijk} * \delta t, \tag{10}$$

where X_{ijk} is a real number representing the bandwidth provided by DC D_i for user cell U_j to download object V_k during the interval δt. So, $X_{ijk} * \delta t$ is the amount of data delivered by D_i to U_j. The communication distance oc_{ij} specifies the unit cost of transferring data from D_i to U_j.

The fourth component, *Uc*, is the update cost arising from uploading and/or updating data object by users:

$$Uc = \sum_{k=1}^{W} \sum_{i=1}^{M} \sum_{j=1}^{N} Uc_{ijk}(Z_{ijk}), \tag{11}$$

$$Uc_{ijk}(Z_{ijk}) = Z_{ijk} * \delta t * (ic_{ij} + \sum_{l=1:Y_{lk}=1\ and\ l \neq i}^{M} dc_{il}), \tag{12}$$

where Z_{ijk} is a real number representing the bandwidth provided by DC D_i for user cell U_j to update object V_k during the interval δt. So, $Z_{ijk} * \delta t$ is the amount of data sent from U_j to D_i. The communication distance ic_{ij} and dc_{il} specify the unit costs of transferring data to D_i from U_j and D_l respectively. The research issue concerning data consistency is beyond the scope of this paper, which we leave as future work. Here we only consider the cost of data updating. The updates to an object V_k are performed by U_j sending the updated data to one of V_k's replicas, which afterwards broadcasts it to every other replica. The first term in Eq. (12) is the cost of sending updated data from U_j to D_i, and the second term is the cost of broadcasting updated data to all other V_k's replicas.

Considering the system architecture described in Section 3.1, we intend to solve the following optimization model:

$$Minimize \quad C_{\delta t} = Sc + Mc + Dc + Uc, \tag{13}$$

Subject to:

$$avail(V_k) \geqslant V_k^{th}, \tag{14}$$

$$\sum_{i=1}^{M} X_{ijk} \geqslant dQ_{jk}, \tag{15}$$

$$\sum_{i=1}^{M} Z_{ijk} \geqslant uQ_{jk}, \tag{16}$$

$$\sum_{k=1}^{W} \sum_{j=1}^{N} X_{ijk} \leqslant dB_i, \ \sum_{k=1}^{W} \sum_{j=1}^{N} Z_{ijk} \leqslant uB_i, \tag{17}$$

$$\sum_{k=1}^{W} v_k * Y_{ik} \leqslant S_i, \tag{18}$$

$$\text{if} \sum_{j=1}^{N} X_{ijk} + Z_{ijk} > 0 \text{ then } Y_{ik} = 1, \tag{19}$$

$$X_{ijk} \geq 0 \; and \; Z_{ijk} \geq 0, \tag{20}$$

$$Y_{ik} \in \{0, 1\}. \tag{21}$$

where $i = 1, \ldots, M, \; j = 1, \ldots, N$ and $k = 1, \ldots, W$.

To guarantee data availability, and the download and upload bandwidth provision, we introduce Constraints 14, 15 and 16 according to the description in Section 3.2. Constraints 17 and 18, which are the capacity constraints, states that the amount of bandwidth and storage assigned from DC D_i cannot exceed its capacity, where dB_i, uB_i and S_i are the total download and upload bandwidth capacity and storage capacity, respectively, of D_i. In Constraint 19, the inequality $\sum_{j=1}^{N} X_{ijk} + Z_{ijk} > 0$ implies that some users need to access a specific object V_k from D_i, so that a replica of V_k must be placed at D_i, i.e., Y_{ik} must be 1. Constraint 20 stipulates that the amount of the reserved bandwidth is a positive value. Constraint 21 stipulates that a replica of object V_k is either placed or not placed in DC D_i.

The proposed optimization problem in this paper can be succinctly stated by its objective function as: *Find the assignments of real values in the* $\boldsymbol{X}$ *and* $\boldsymbol{Z}$ *matrices and of Boolean values {0, 1} in the*$\boldsymbol{Y}$ *matrix that are related to* $\boldsymbol{X}$ *and* $\boldsymbol{Z}$*, so that the objective function* 13 *is minimized subject to the Constraints 14-21, where* $i = 1, \ldots, M, \; j = 1, \ldots, N$ *and* $k = 1, \ldots, W$.

3.4 Analysis and Transformation of the Model

Theorem 1. *The minimization problem described by Eq. (13) and Constraints 14-21 is NP-hard.*

Proof. Let's first make the following three assumptions: (1) there are no accesses from users to objects so that all elements of X and Z matrices 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 data object. Based on these assumptions, all the terms of Eq. (13), except for the first term, and Constraints 15, 16, 17, 19 and 20 can be removed. So the original problem is reduced to the following simplified minimization sub-problem:

$$minC_{\delta t} = \sum_{k=1}^{W} \sum_{i=1:Y_{ik}=1}^{M} k_i * v_k * \delta t, \tag{22}$$

subject to Constraints 18, 21 and

$$\sum_{i=1}^{M} Y_{ik} \geq 1, k = 1, \ldots, W. \tag{23}$$

Obviously, this sub-problem is a minimization version of the generalized assignment problem, which Chekuri and Khanna [26] 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 14, 15 and 16, we introduce three penalty functions. If any demand for bandwidth or availability cannot be met, the objective function value of Eq. (13) will be increased by a big number depending 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 14, the penalty function term is $p_k^1 = P_1 * (V_k^{th} - avail(V_k)) * v_k$, when $V_k^{th} > avail(V_k)$, otherwise $p_k^1 = 0$. With regard to Constraint 15, the penalty function term is $p_{jk}^2 = P_2 * (dQ_{jk} - \sum_{i=1}^{M} X_{ijk})$, when $dQ_{jk} > \sum_{i=1}^{M} X_{ijk}$, otherwise $p_{jk}^2 = 0$. And with regard to Constraint 16, the penalty function term is $p_{jk}^3 = P_3 * (uQ_{jk} - \sum_{i=1}^{M} Z_{ijk})$, when $uQ_{jk} > \sum_{i=1}^{M} Z_{ijk}$, otherwise $p_{jk}^3 = 0$. As penalty factors, all P_1, P_2 and P_3 are real numbers. So we get a new equivalent model with Constraints 17-21 and a new objective function, which is:

$$minC_{\delta t} = Sc + Mc + Dc + Uc + \sum_{k=1}^{W} \left\{ p_k^1 + \sum_{j=1}^{N} (p_{jk}^2 + p_{jk}^3) \right\}. \tag{24}$$

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 objects and allocating bandwidth. From Eq. (24) 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 adjusting the three penalty factors and then define the virtual gain of resources assignment as difference value between them. Thus, the equivalent maximization problem is:

$$\begin{aligned} max & \left(\sum_{k=1}^{W} q_k^1 - Sc - Mc \right) \\ & + \left(\sum_{k=1}^{W} \sum_{j=1}^{N} q_{jk}^2 - Dc \right) + \left(\sum_{k=1}^{W} \sum_{j=1}^{N} q_{jk}^3 - Uc \right), \end{aligned} \tag{25}$$

subject to Constraints 17-21, where $q_k^1 = P_1 * avail(V_k)) * v_k$ when $V_k^{th} > avail(V_k))$, otherwise $q_k^1 = P_1 * V_k^{th} * v_k$, $q_{jk}^2 = P_2 * \sum_{i=1}^{M} X_{ijk}$ when $dQ_{jk} > \sum_{i=1}^{M} X_{ijk}$, otherwise $q_{jk}^2 = P_2 * dQ_{jk}$, and $q_{jk}^3 = P_3 * \sum_{i=1}^{M} Z_{ijk}$ when $uQ_{jk} > \sum_{i=1}^{M} Z_{ijk}$, otherwise $q_{jk}^3 = P_3 * uQ_{jk}$.

The first term of Eq. (25) represents the virtual gain obtained by ensuring the V_k's availability, while the second and the third terms represent the additional gain obtained by maintaining bandwidth provision for user cell U_j accessing data object 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 objects if no resources are assigned $\{\sum_{k=1}^{W} P_1 * V_k^{th} * v_k + \sum_{k=1}^{W} \sum_{j=1}^{N} (P_2 * dQ_{jk} + P_3 * uQ_{jk})\}$ and the operating cost obtained by resources assignment given by the objective function in Eq. (13).

4 Distributed Resource Scheduling Algorithm

Because the optimization problem is NP-hard, in this section, we propose a distributed heuristic algorithm for replica placement and bandwidth allocation, called *DREAM*, which solves the optimization problem described by objective function 25 and Constraints 17-21. The *DREAM* algorithm for resource scheduling is executed by each DC within the cloud platform.

4.1 Preliminaries

The algorithm determines the placement of object replicas and bandwidth assignment based on the ratio between the resource-assignment gain and the cost described by Eq. (25) and Eq. (13) respectively. The algorithm has as input four parameters, $\boldsymbol{dQ}$, $\boldsymbol{uQ}$, $\boldsymbol{V^{th}}$ and $\boldsymbol{Y_i'}$. The first two parameters $\boldsymbol{dQ}$ and $\boldsymbol{uQ}$ are the $N * W$ matrixes of N user cells' download and upload bandwidth demands respectively for W objects, which are derived by prediction or collection in advance. The third parameter $\boldsymbol{V^{th}}$ is a W-element vector $(V_1^{th}, V_2^{th}, \ldots, V_W^{th})$ of which each entry is the threshold of an object's availability. The last parameter $\boldsymbol{Y_i'}$ is a W-element vector of which each entry Y_{ik}', $k = 1, \ldots, W$, is equal to one when the object V_k is already replicated in DC D_i by the last execution of *DREAM*, otherwise it is zero.

To help describe *DREAM*, now we calculate the operating cost and virtual assignment gain for a tuple consisting of DC D_i, user cell U_j and object V_k. According to Eq. (6), (8), (10) and (12), the operating cost of resource (storage space and bandwidth) assignment, rc_{ijk} $(i = 1, \ldots, M$, $j = 1, \ldots, N$ and $k = 1, \ldots, W)$, is defined as follows:

$$rc_{i0k} = (Sc_{ik} + Mc_{ik}) * (1 - Y_{ik}), \tag{26}$$

$$rc_{ijk} = Dc_{ijk}(x_{ijk}) + Uc_{ijk}(z_{ijk}) + rc_{i0k}, \tag{27}$$

where $x_{ijk} = min(rdQ_{jk}, rdB_i)$ and $z_{ijk} = min(ruQ_{jk}, ruB_i)$, rdQ_{jk} and ruQ_{jk} record U_j's remaining download and upload bandwidth demands for V_k, rdB_i and ruB_i record the quantity of DC D_i's remaining upload and download bandwidth capacities. So x_{ijk} and z_{ijk} record the part of the remaining bandwidth demands, which can be met by D_i. Clearly, rc_{i0k} represents the cost of placing an object V_k's replica in D_i, including storage cost, Sc_{ik}, and migration cost, Mc_{ik}. rc_{ijk} represents the sum of the cost the application will experience if D_i provides x_{ijk} download bandwidth and z_{ijk} upload bandwidth to user cell U_j to access V_k. The first two terms of Eq. (27), $Dc_{ijk}(x_{ijk})$ and $Uc_{ijk}(z_{ijk})$, are the delivery cost and the update cost respectively. Sc_{ik}, Mc_{ik}, $Dc_{ijk}(x_{ijk})$ and $Uc_{ijk}(z_{ijk})$ are computed according to Eq. (6), (8), (10) and (12).

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

$$rg_{i0k} = \begin{cases} (P_1 * y_{ik} * v_k - rc_{i0k})(1 - Y_{ik}) \\ \qquad\qquad \text{if } avail(V_k) < V_k^{th} \\ -rc_{i0k} \\ \qquad\qquad \text{if } avail(V_k) \geq V_k^{th}, \end{cases} \tag{28}$$

if $avail(V_k) < V_k^{th}, y_{ik} = min(V_k^{th} - avail(V_k), davail(D_i, V_k))$, where $davail(D_i, V_k)$ returns the growth value of V_k's availability if V_k's replica is placed in D_i, otherwise $y_{ik} = 0$. The term $avail(V_k) \geq V_k^{th}$ in Eq. (28) means that 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. As can be seen from the definition of rg_{i0k}, it represents the increase in overall gain the bandwidth-intensive application would experience if D_i replicates V_k.

$$rg_{ijk} = (P_2 * x_{ijk} - Dc_{ijk}(x_{ijk})) \\ + (P_3 * z_{ijk} - Uc_{ijk}(z_{ijk})) + rg_{i0k}. \tag{29}$$

rg_{ijk} represents the sum of gain in the application when bandwidth x_{ijk} and z_{ijk} are allocated to users. The first two terms $P_2 * x_{ijk} - Dc_{ijk}(x_{ijk})$ and $P_3 * z_{ijk} - Uc_{ijk}(z_{ijk})$ in Eq. (29) indicate the gain of meeting download and upload bandwidth demands. And the third term indicates the benefit of meeting V_k's availability demand, if V_k is replicated at the same time. Of course, we will set P_1, P_2 and P_3 great enough to make rg_{i0k} and rg_{ijk} positive numbers when higher availability and more bandwidth are required for the object V_k and user cell U_j.

In what follows we define the cost-effectiveness of allocating resources, ce_{ijk}, which is the ratio between the virtual resource assignment gain and the corresponding cost $(i = 1, \ldots, M$, $j = 1, \ldots, N$ and $k = 1, \ldots, W)$:

$$ce_{ijk} = rg_{ijk}/rc_{ijk}. \tag{30}$$

The cost-effectiveness measure ce_{ijk} is used to determine each "global" decision of replicating data objects and/or assigning bandwidth. In making these decisions, *DREAM* considers the effect of allocating the resources on the overall application gain and cost. *DREAM* always assigns resources that attain the global maximum cost-effectiveness value ce_{max}, until all ce_{ijk}s are non-positive. According to Eqs. (27), (29), and (30), some of ce_{ijk}s are recalculated after every resource assignment by *DREAM* and they will be non-positive if the demands are met completely or there are no more resources that can be assigned in corresponding DCs.

4.2 The Proposed Algorithm

DREAM starts with an initialization phase in which $D_i (i = 1, \ldots, M)$ initializes its local variables, including the cost-effectiveness matrix $\boldsymbol{ce_i}$ calculated by Eq. (30) (Line 2).

The second phase of the algorithm is iterative in nature, consisting of the while loop depicted in Lines 5-14. Before entering the loop, each DC D_i determines its local highest cost-effectiveness value $ce_{max}(i)$. And then all DCs participate in a collective communication operation called *all-reduce-max* [24] (Line 3) to determine the global maximum cost-effectiveness value, ce_{max}. In each iteration, if DC D_i originates ce_{max}, it performs replica placement and bandwidth allocation if needed (Line 7). If D_i does not attain the maximum global cost-effectiveness value, it only updates some local values to keep track of the changes resulting from allocation of bandwidth and storage space at other DCs (Line 10). According to Eq. (26)-(30), some cost-effectiveness values will change and need to be recalculated after every resource assignment. Therefore, each DC D_i updates its cost-effectiveness matrix $\boldsymbol{ce_i}$ (Line 12). In Line 13, each DC participates in another all-reduce-max operation that determines the next candidate maximum cost-effectiveness value ce_{max}. All ce_{ijk}s' values will be non-positive if all demands for bandwidth and availability are met or there are no more resources that can be assigned in the corresponding DCs. At this point, the algorithm will terminate.

Algorithm 1. DREAM($\boldsymbol{dQ}, \boldsymbol{uQ}, \boldsymbol{V^{th}}, \boldsymbol{Y_i^{t-1}}$)

```
1: Initialization of data center $D_i$, $i = 1, \dots, M$.
2: $D_i$ computes its cost-effectiveness matrix $\boldsymbol{ce_i}$ according to
   Eq. (30) and picks out the maximum element $ce_{max}(i)$.
3: $D_i$ finds out the global maximum cost-effectiveness value
   $ce_{max}$, corresponding to a specific pair of user cell $U_j$ and
   object $V_k$, from all $ce_{max}(i)$, $i = 1, \dots, M$, by participating in
   the distributed procedure **all-reduce-max** [17] with all other
   DCs.
4: Iteration of DC $D_i$, $i = 1, \dots, M$.
5: **while** $ce_{max} > 0$ **do**
6:     **if** this DC $D_i$ generates the maximum cost-effectiveness
       value $ce_{max}$ **then**
7:         $D_i$ either assigns download and upload bandwidths,
           $x_{ijk}$ and $z_{ijk}$ in Eq. (27), to user cell $U_j$ to access data
           object $V_k$, replicates $V_k$ in $D_i$, or does both.
8:     **else**
9:         {another DC has the maximum cost-effectiveness value
           $ce_{max}$}
10:        Updates some local values to keep track of the changes
           resulting from allocation of bandwidth and storage
           space.
11:    **end if**
12:    $D_i$ updates its cost-effectiveness matrix $\boldsymbol{ce_i}$ according to
       Eq. (30) and picks out the maximum element to prepare
       for the next iteration.
13:    As in Line 3, $D_i$ finds out the global maximum cost-effec-
       tiveness value $ce_{max}$ by participating in the procedure
       **all-reduce-max**.
14: **end while**
```

4.3 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 *download bandwidth allocating iteration* is one that only allocates download bandwidth to a user cell, and similarly an *upload bandwidth allocating iteration* is one that only allocates upload bandwidth to a user cell. The two types of bandwidth allocating iteration mean that no object's replica is placed in this iteration. A *replica placement iteration* is one that only places an object'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 cell from this replica as a *hybrid iteration*. It is clear from the algorithm description (Lines 6-11) that each iteration falls into one of these four categories. Moreover, Eq. (30) shows 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 and z_{ijk}s of a pair of a DC and a data object are equal to zero.

Based on the analysis above, it is not difficult to infer that a data object may introduce at most $N-1$ *download bandwidth allocating iterations* and $N-1$ *upload bandwidth allocating iterations*. Thus, for all data objects the maximum number of *bandwidth allocating iterations* is $W*2*(N-1)$. If sufficient storage space is provided, in the worst case, an object 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-case computing time of each iteration is $W*N+log(W*N)$. Thus the computing time of the main loop is $W*(M+2N-2)*(W*N+log(W*N))$. The initialization phase consists of a $(W*N)$-iteration loop (Line 2) and a *build-heap* operation of cost $(W*N)$. Therefore, the worst case computing 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 cells, and W is the total number of objects. According to the literature [24], it is easy to prove that the communication complexity of *DREAM* is $\boldsymbol{O(W*(M+N)*logM)}$. So, both types of complexity are polynomial. Please note that these two types of complexity analyzed above are the worst case complexities, in the other cases the computing time and the communication complexity are also related to the users' demands and the amount of cloud resource. For example, if there is no users' demands or there is no cloud resource at all, obviously the two types of complexity will be constants.

4.4 Integrating Locality-Awareness

To achieve high access locality, some modifications need to be made to the *DREAM* algorithm. First, we define two types of virtual communication distance for transferring data units out of and into DC D_i by two floating-point value functions voc_{ij} and vic_{ij} as:

$$voc_{ij} = \begin{cases} oc_{ij} & \text{if } D_i \text{ and } U_j \text{ are in the same region,} \\ oc_{ij} + P_4 & \text{if } D_i \text{ and } U_j \text{ are in different regions,} \end{cases}$$

$$vic_{ij} = \begin{cases} ic_{ij} & \text{if } D_i \text{ and } U_j \text{ are in the same region,} \\ ic_{ij} + P_5 & \text{if } D_i \text{ and } U_j \text{ are in different regions,} \end{cases}$$

where $i = 1, \dots, M$ and $j = 1, \dots, N$. P_4 and P_5, referred to as the locality penalty factors, are two positive real numbers. Second, we replace oc_{ij} and ic_{ij} with voc_{ij} and vic_{ij} in Eq. (10) and (12) respectively. From Eq. (30), we can see that *DREAM* tends to allocate bandwidth to users from DCs with smaller communication distance. We refer to the *DREAM* algorithm integrated with locality-awareness as *DREAM-L*.

4.5 Object Grouping

While Section 4.3 shows that the running time of *DREAM* is not exponential in complexity, like the majority of the existing algorithms solving a constraint optimization problem [13], it nonetheless still suffers from long running time, typically in the order of several or tens of thousand seconds, when the number of data objects is large. Therefore, we propose and incorporate an object-grouping technique into *DREAM* and *DREAM-L*, called *DREAM-G* and *DREAM-LG* respectively, which helps reduce the running time to a small time range at a small and acceptable cost of scheduling quality as demonstrated in Section 5.4.

The object-grouping technique described in Algorithm 2 entails the following steps. In Line 1, we set an upper bound on the size of a data object group gs, say, 128 data objects, and then randomly divide all objects into groups. Thus each group has gs data objects except for the last one that may have fewer than gs data objects. In Lines 3 and 4, we calculate each object group's demands for bandwidth and availability and, based on the proportions of demands among the object groups, assign DCs' available resources (bandwidth and storage space) to all the object groups. For example, assume that (1) there are two DCs in the cloud, D_1 and D_2, which have 3 and 6 Mbps bandwidth capacity for delivering data respectively, and (2) two object groups, R_1 and R_2, from which all users need 2 and 4 Mbps bandwidth to download data respectively. According to Line 4, D_1 assigns 1 and 2 Mbps to R_1 and R_2 respectively, and D_2 assigns 2 and 4 Mbps to R_1 and R_2 respectively. After that Algorithm 2 assigns cloud's bandwidth capacity for receiving data and storage space to all object groups in the same way. In addition, before assigning DSs' storage space, Algorithm 2 should first estimate how many byte each group will store according to their object availability demands. Then, in Line 5, *DREAM* (or *DREAM-L*) is executed in all the object groups in parallel. After the execution of the optimization algorithm for each group, in Line 6, each DC executes *DREAM(-L)*($\boldsymbol{dQ}$, $\boldsymbol{uQ}$, $\boldsymbol{V^{th}}$, $\boldsymbol{Y_i'}$), which uses the equation $ce_{ijk} = rg_{i0k}/rc_{i0k}$ instead of Eq. (30) to calculate ce_{ijk}s ($i = 1, \ldots, M, j = 1, \ldots, N$ and $k = 1, \ldots, W$) for meeting the unmet data availability.

Algorithm 2. DREAM-G(gs, $\boldsymbol{dQ}$, $\boldsymbol{uQ}$, $\boldsymbol{V^{th}}$, $\boldsymbol{Y_i'}$)

```
1: Randomly divide all data objects into groups, each of which has at most $gs$ objects, given that there are $H = \lceil W/gs \rceil$ data object groups $R_1, \ldots R_H$, where $W$ is the total number of objects.
2: Inform all DCs of all object identifiers of each object group.
3: Each DC calculates each object group's demands, for download bandwidth $\boldsymbol{dQ(R_h)}$, upload bandwidth $\boldsymbol{uQ(R_h)}$ and availability $\boldsymbol{V^{th}(R_h)}$, $h = 1, \ldots, H$ respectively.
4: Based on the proportions of demands among the object groups, each DC assigns its available bandwidth and storage space to all the data object groups.
5: Each DC executes the algorithm DREAM(-L)($\boldsymbol{dQ(R_h)}$, $\boldsymbol{uQ(R_h)}$, $\boldsymbol{V^{th}(R_h)}$, $\boldsymbol{Y_i'(R_h)}$) in all the object groups in parallel.
6: Execute the algorithm DREAM(-L)($\boldsymbol{dQ}$, $\boldsymbol{uQ}$, $\boldsymbol{V^{th}}$, $\boldsymbol{Y_i'}$) only for the unmet data availability.
```

Given that most of or even all objects' availability is met in Line 5, this step (Line 6) requires very little or no computation. Based on the analysis in Section 4.3, it is easy to derive the conclusion that the worst-case computing time and communication complexity of *DREAM(-L)* in each object group are $O((M+N)*gs^2*N)$ and $O(gs*(M+N)*logM)$ respectively. On the one hand, if each DC has only one CPU with one core for all groups' computing, all groups will in turn run *DREAM(-L)* and the worst-case computing time of the new *DREAM-LG* will be $(W/gs)*O((M+N)*gs^2*N) = O((M+N)*W*gs*N)$, where (W/gs) is the total number of object groups. On the other hand, if each DC has (W/gs) processors for all (W/gs) groups to run *DREAM(-L)*, it means that each group has one processor in each DC. As a result, *DREAM-LG* in all object groups are executed in parallel, leading to the worst-case computing time of *DREAM-LG* will be $(W/gs)*O((M+N)*gs^2*N)/(W/gs) = \boldsymbol{O((M+N)*gs^2*N)}$, and the corresponding communication complexity of $\boldsymbol{O(gs*(M+N)*logM)}$. Of course, when the number of processors is more than one but fewer than (W/gs), the computing time will fall somewhere between the two cases. Given the ubiquitous use of multi-core processors and GPU and increasing popularity of cloud computing, it is reasonably justified for us to assume in this paper that there is sufficient compute resource on each DC so that the number of processors running DREAM-LG in each DC is at least (W/gs), which leads to the aforementioned computing time and communication complexity. This means that when M and N are fixed, the running time of *DREAM-LG* depends on gs, the size of a data object group, and is independent of W, where W is the number of objects, N is the number of user cells, and M is the number of DCs. Therefore, even when the number of data objects scales and W becomes large, because we set gs fixed, the running time of *DREAM-LG* still falls within an acceptable range, which is about 1-50 seconds as experiments in Section 5.4 show.

5 Evaluations on DREAM-LG Versus Decision Tree

In this section, we perform simulation experiments to evaluate the performance of *DREAM-LG*. We compare the performance of *DREAM-LG* with that of the online algorithm *Decision Tree* [3]. We choose to compare *DREAM-LG* against the *DT* algorithm because it is, to the best of our knowledge, the closest work to ours in that it has the similar optimization targets, including data availability, access locality, cost-effectiveness and load assignment, and application environment in the cloud platform. We focus on comparing the performance of the two algorithms in a range of cases with different load distribution patterns, system capacities, and numbers of DCs, user cells and data objects, in order to study the performance impact of these systems and workload parameters.

In our experiments, all DCs are assumed to be uniformly distributed in all regions. We further assume that 1) as other researcher's work [3], all DCs have the same confidence level ($conf_i = 0.98, i = 1, \ldots, M$ where M is the number of DCs) , and 2) as general cases, each data object needs 2 or 3 replicas in different DCs to maintain its availability. We set the interval between two runs of our optimization algorithm to be $\delta t = 10$ minutes, and the storage space price and data transfer price based on the pricing policies of Amazon's simple storage service (S3) [5] as of July 2013.

5.1 Evaluation Metrics

We use the following 10 metrics in our simulation, three for measuring the quality of service, two for specifying system configurations, and five for estimating costs. According Section 3.2, the three QoS metrics are *Service availability*, the arithmetic mean of the ratios of all data objects' availability to their availability thresholds during a 10-minute interval, *Service bandwidth provision*, the arithmetic mean of all user cells' bandwidth provision, and *Service access locality*, the arithmetic mean of all user cells' access locality. The two system configuration metrics are *Replication degree*, the average number of replicas per object and *Cross-region traffic*, the amount of traffic that crosses different regions. The five cost metrics, *Delivery cost* (Dc), *Update cost* (Uc), *Storage cost* (Sc), *Migration cost* (Mc) and *Operating cost* (Oc), have been defined formally in Section 3.3. Informally, *Delivery cost* is the communication cost of delivering data from the cloud to users, *Update cost* is the communication cost of updating

TABLE 3
A Comparison Between DREAM-LG and DT in Terms of the Mean Values of the Key QoS Measures with Different Bandwidth Capacity

Capacity	**Data Availability**			**Bandwidth Provision**			**Access Locality**		
/ Demand	***DREAM-LG /DT***	***DREAM-LG***	***DT***	***DREAM-LG /DT***	***DREAM-LG***	***DT***	***DREAM-LG /DT***	***DREAM-LG***	***DT***
1	100%	100%	100%	194.1%	99.7%	51.3%	122.9%	61.6%	50.1%
2	100%	100%	100%	150.2%	99.9%	66.5%	164.4%	74.8%	45.5%
4	100%	100%	100%	126.2%	99.9%	79.2%	206%	85.6%	41.6%
8	100%	100%	100%	111.7%	99.9%	89.4%	240.6%	92.8%	38.5%

data, and *Migration cost* is the communication cost of moving data among DCs; *Storage cost* is the cost of storing data in the cloud. *Operating cost* is the sum of *Delivery cost*, *Update cost*, *Storage cost* and *Migration cost*. Note that all costs, in *US dollars*, are derived based on the similar pricing scales of Amazon's S3 as of July 2013.

5.2 Setup

In the first set of experiments, we compare the performance of *DREAM-LG* against that of *Decision Tree* for a wide range of scenarios with different system configurations and different distributions of bandwidth demand. As mentioned above, W, M, and N are the total numbers of data objects, DCs and user cells respectively. We consider the following scenarios: $W = 1{,}024, 8{,}192, 16{,}384$ with ($M = 64$ and $N = 64$), ($M = 128$ and $N = 64$), ($M = 128$ and $N = 128$) and ($M = 256$ and $N = 128$) respectively. For each (W, M, N) combination, the available bandwidth capacity of the entire cloud is 1X, 2X, 4X and 8X of the total bandwidth demand. Each object group has 128 data objects.

We compare the performance of *DREAM-LG* and *DT* for many different types of bandwidth demand distributions in each scenario with a (W, M, N) combination and a type of cloud bandwidth capacity. These distributions could fall into uniform distribution in which bandwidth demands are distributed equally among data objects and users or skewed distribution in which data objects and users assume different amounts of bandwidth demands.

Let bandwidth demand of user cell U_j for accessing data object V_k be Q_{jk}. Based on the Pareto Principle, we set U_j's download and upload bandwidth demands for V_k, dQ_{jk} and uQ_{jk}, that account for 80 and 20 percent of Q_{jk} respectively. Q_{jk} ($j = 1, \ldots, N$ and $k = 1, \ldots, W$) is defined as follows:

$$Q_{jk} = (t_{jk} * total\ bandwidth\ demand) / \sum_{j=1}^{N} \sum_{k=1}^{W} t_{jk}, \qquad (31)$$

where t_{jk} is called the popularity of user cell U_j for object V_k, which represents the distribution of bandwidth demand of U_j over all objects and is derived by a set of three parameters similar to those used in [12]. The first parameter, which is a floating point number and called the hot-set parameter θ, determines the distribution of the popularity of any user cell, say U_j, among all objects as $t_{jk} = e^{(-\theta k)}/S$, where $k = 1, \ldots, W$ and $S = \sum_{k=1}^{W} e^{-\theta k}$.

The second parameter, called the correlation of user cell hot-set, ρ, determines the randomness of popularity of an object for different user cells. It can be characterized as follows. First, we determine the popularity for user cell U_1 without using ρ. Therefore, U_1 has the kth lowest popularity for object V_k, based on equation $t_{1k} = e^{(-\theta k)}/S$. Now, at a user cell other than U_1, object V_k will have the k'th lowest popularity among the objects, where k' is a random number between 1 and $\min(\rho * W + k - 1, W)$. The third parameter is the relative user cell activity, η. After we determine the popularity for different combinations of user cells and data objects with the above two parameters, we multiply all the popularity t_{jk}s at user cell U_j by $1/(A * j^{(1-\eta)})$, where $A = \sum_{j=1}^{N} 1/j^{(1-\eta)}, 0 \leqslant \eta \leqslant 1$ and $k = 1, \ldots, W$.

In this set of experiments, with the increments of 0.018, 0.1 and 0.2, we vary θ from 0.01 to 0.208, ρ from 0.06 to 0.96, and η from 0 to 1 respectively. So the data set of the experiments, which results from 3*4*4*12*10*6=34,560 experiments, is a very large set.

5.3 Comparing against Decision Tree

In this set of experiments, we focus on comparing the performances of *DREAM-LG* and *DT* and examining how the variability of system configurations and distribution of bandwidth demands affect their performances. Hence, we select a wide range of parameter values. In Tables 3 and 4, we show the statistics of the QoS measurement and cost metrics when considering different scenarios of bandwidth provision. Each scenario includes all cases that have parameters $(W, M, N, \theta, \rho, \eta)$ different from one another. The first column in the two tables shows the ratio of total cloud bandwidth capacity to total bandwidth demand.

From Tables 3 and 4, we can see that when the available bandwidth capacity of the entire cloud platform is 4× or 8× of the total bandwidth demand, which means that when the bandwidth resource is highly abundant in the cloud, our *DREAM-LG* improves over *DT* in bandwidth provision

TABLE 4
DREAM-LG's Mean Values of Costs and Cross-Region Traffic Normalized to Those of DT as a Function of Total Cloud Bandwidth Capacity

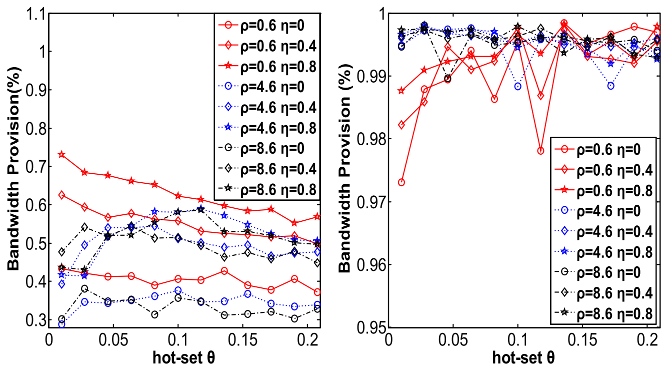
Capacity	**DREAM-LG / DT**					
/Demand	***Sc***	***Mc***	***Dc***	***Uc***	***Oc***	***Crt***
1	39%	118%	169%	79.1%	154%	168%
2	35%	99%	110%	40.4%	103%	98.1%
4	34%	91%	94.9%	25.5%	88.3%	56.6%
8	33%	88%	90.5%	18.6%	83.5%	31.9%

Sc: Storage cost, Mc: Migration cost, Dc: Delivery cost, Uc: Update cost, Oc: Operating cost = Sc+Mc+Dc+Uc, and Crt: Cross-region traffic.

TABLE 5
The Results of a Subset of Experiments in Which We Force *DREAM-LG* to Reach the Same Level of Bandwidth Provision as *DT* by Limiting Quantity of Bandwidth it Offered

Capacity /Demand	DREAM-LG / DT			
	Oc	*Bdth*	*Local*	*Avail*
1	87.0%	101%	116%	100%
2	72.15%	101%	163%	100%
4	71.93%	100%	206%	100%
8	75.57%	100%	241%	100%

Oc: *Operating cost,* Bdth: *Bandwidth Provision,* Local: *Access locality, and* Avail: *Data availability.*



(a) *DT* (b) *DREAM-LG*

Fig. 2. The bandwidth quantity curves as a function of the distributions of users' bandwidth demands.

and access locality by 11.7-26.2 and 106-140.6 percent respectively, while saving 11.7-16.5 percent operating costs. On the other hand, when the available bandwidth capacity of the entire cloud is less abundant, at only 1× or 2× of the total bandwidth demand, our *DREAM-LG*'s advantage over *DT* in bandwidth provision and access locality remains significant, at 50.2-94.1 and 22.9-64.4 percent respectively, at only 2.9-54.4 percent additional operating costs because of more cloud bandwidth being provided.

We argue that when the bandwidth resource is tight, the additional operating cost of 2.9-54.4 percent incurred by *DREAM-LG* is acceptable based on the two following reasons. First, *DREAM-LG* is more cost-effective than *DT*. As shown in Table 5, when we force *DREAM-LG* to reach the same level of bandwidth provision as *DT* by limiting quantity of bandwidth it offered, *DREAM-LG* save 13-28 percent of the operating cost of *DT* and still offer higher access locality. Second, *DREAM-LG* has a strong capability of providing sufficient bandwidth to meet users' demands and thus it is more applicable to bandwidth intensive applications than *DT*. Because generally the more the users' demands are met, the more the application providers earn. That's the root reason why application providers purchase cloud resources to run their applications.

In this set of experiments, we find that the lack of a global bandwidth scheduling scheme in *DT* result in poor bandwidth provision due to DCs' bandwidth overload. In particular, when the available bandwidth capacity of the entire cloud is exactly the same as the total bandwidth demand, the *DT* algorithm is only able to achieve an average of 50.1 percent bandwidth provision. It is not until the available bandwidth of the entire cloud is up to 8× of the total bandwidth demand, can the *DT* algorithm achieve an average of 89.4 percent bandwidth provision, which still is lower than our *DREAM-LG*. However, given the fact that the bandwidth resource is still an expensive and scarce resource in the current Internet, it is arguably unrealistic to guarantee the available bandwidth to be as much as up to 8 times that of an application's bandwidth demand. In contrast, our *DREAM-LG* is able to achieve almost all (99.7 percent) bandwidth provision even when the available bandwidth is 1× of the total bandwidth demand. And with an increasing amount of available bandwidth, bandwidth provision achieved by *DREAM-LG* will be even higher.

As shown in Table 3, *DT* provides not only just 38.5-50.1 percent access locality but also a decreasing access locality as the available bandwidth increases, which means that the users must obtain more bandwidth from the remote regions to meet their increasing bandwidth demand. In comparison, our *DREAM-LG* provides access locality that is not only 1.2-2.4 times higher than that by *DT*, but also increases with the available bandwidth in the cloud. The much higher access locality provided by *DREAM-LG* than by *DT* means that the former utilizes much more local bandwidth resources to serve the users so as to reduce the cross-region traffic than the latter. Because the cost and thus price of inter-region traffic is always higher than that of intra-region traffic, reducing cross-region traffic decreases the overhead of the cloud. Meanwhile, the workload of the backbone network will also be cut down.

We also observe that the variability of bandwidth provision of *DREAM-LG* and *DT* exhibits some general trends that are closely associated with the distributions of bandwidth demands. We examine the case illustrated in Fig. 2 as an example to demonstrate these trends. Fig. 2 shows the bandwidth provision curves as a function of the distributions of users' bandwidth demands when the (W, M, N) combination is (16,384, 256, 128) and the available bandwidth is 1× of the total bandwidth demand. From Fig. 2a we can see that the performance of *DT* in bandwidth provision fluctuates significantly from 28.87 to 73.08 percent as a function of the distributions of user cells' bandwidth demand. In contrast to *DT*, *DREAM-LG* is shown in Fig. 2b to be almost oblivious to the changes in the distribution of bandwidth demand, achieving consistently high performance in bandwidth provision at above 97.31 percent for all kinds of distribution of bandwidth demand.

5.4 Running Time

Table 6 reports the running time and QoS measurement for some chosen problem instances in which *DREAM-L* was terminated in a reasonable amount of time.

By analyzing the experimental results as shown in Table 6, we can see that by using the object-grouping technique, *DREAM-LG* substantially reduces the running time, without either increasing the operating cost significantly or reducing the QoS. When the bandwidth resource is tight, *DREAM-LG* has worse locality than *DREAM-L*, but when more cloud bandwidth is provided, they have similar levels of locality. It may be argued that, for higher computational efficiency, in some situations it is necessary and inevitable to weaken the

TABLE 6
A Comparison Between DREAM-LG and DREAM-L in Terms of the Mean Values of the Running Time, Operating Cost and QoS Measures

Capacity /Demand	*W*	DREAM-LG *Time*	DREAM-L *Time*	DREAM-LG / DREAM-L *Oc*	*Bdth*	*Local*	*Avail*
1	1,024	9.15	503	99.4%	99.7%	84.0%	100%
	2,048	9.57	2,143	101%	99.8%	76.5%	100%
2	1,024	8.73	517	101%	100%	93.3%	100%
	2,048	9.02	2,311	101%	100%	89.1%	100%
4	1,024	9.68	621	99.3%	100%	96.8%	100%
	2,048	9.64	3,069	99.1%	100%	95.7%	100%
8	1,024	8.95	536	98.7%	100%	98.0%	100%
	2,048	11.3	2,379	96.2%	100%	97.9%	100%

W: *Total number of objects,* M=128: *Total number of DCs,* N=64: *Total number of user cells,* Time: *Average running time,* Oc: *Operating cost,* Bdth: *Bandwidth Provision,* Local: *Access locality, and* Avail: *Data availability. For each (W, M, N) combination, with the increments of 0.018, 0.1 and 0.2, we vary θ from 0.01 to 0.208, ρ from 0.06 to 0.96, and η from 0 to 1 respectively as described in Section 5.2, so each combination includes 720 experiments.*

algorithm's performance. Moreover, *DREAM-LG* still has better locality than the state-of-the-art algorithms, i.e., *DT*, *Per-DC Lim* and *Optimal Load Direction* Note that the experiment results of *Per-DC Lim* and *Optimal Load Direction* will be presented in next section to provide more details.

Fig. 3 shows that the total execution time of *DREAM-LG* does not increase with the total number of the data objects even when the number reaches several hundreds of thousands, instead it decreases with the number of data objects. The reason behind this counterintuitive phenomenon is explained as follows. As the total number of objects increases, the bandwidth load is distributed to a larger number of object groups, which means that for the same number of user cells the bandwidth demand on each object group decreases. This leads to a smaller amount of computation for each data object group, thus resulting in shorter execution time. However, as the number of data objects increases, as shown in Fig. 3, the execution time of *DREAM-LG* initially decreases and then gradually stabilizes to a small range, which is about 1-50 seconds. Moreover, according to Section 4.3, the execution time of the *DREAM-LG* algorithm will grow with the numbers of DCs *M* and user cells *N*. So as indicated in Fig. 3, the larger the *M* and *N* values are, the higher their corresponding curves are. On the other hand, the execution time on each data object group is dependent on the group size when *M* and *N* are fixed according to Section 4.5. Consequently, Fig. 3 shows that a higher curve corresponds to data object groups of larger sizes.

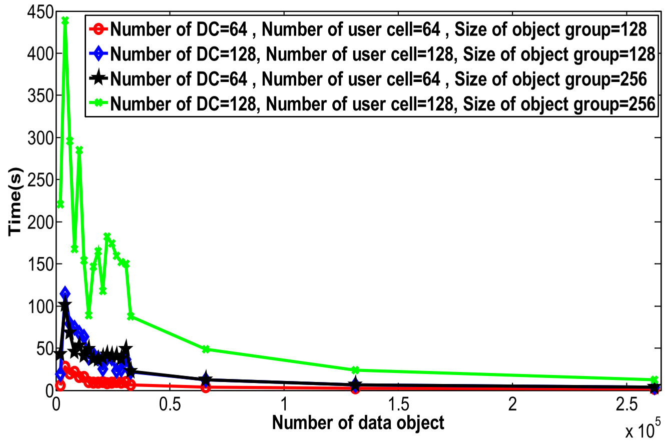


Fig. 3. The running time of *DREAM-LG* as a function of the number of data objects.

6 Evaluation on VoD Application

This set of experiments evaluate the effectiveness of *DREAM-(L)G* on one of the most representative bandwidth-intensive applications, the VoD (video on demand) application, driven by real traces collected from a popular VoD site [25]. In this set of experiments, we compare *DREAM-(L)G* and two latest VoD-specific algorithms, *Per-DC Lim* and *Optimal Load Direction* [4], and demonstrate that our optimization algorithms are superior to the state-of-the-art resource-scheduling algorithms.

6.1 Setup

To evaluate the effectiveness of *DREAM-G* and *DREAM-LG* on the VoD applications, we conduct simulation experiments that are driven by real-world traces collected from the Youku [25] site (a large Chinese commercial VoD site and China's equivalent of YouTube), which covers a 10-day duration. We randomly chose more than 2,000 videos and obtained their statistical information using two online crawlers at every 10 minutes. The obtained information includes videos' durations, the total number of online video replays and and the distribution of the number of online video replays among all Chinese regions. The first crawler collected the information of 1,715 videos that had already been launched for a few days. The second crawler collected traces of the accumulated videos launched in subsequent days at a rate of 38-40 videos per day.

To assign and reserve sufficient bandwidth for users' requests by executing the optimization algorithm in advance, taking the historical traces as input, we implement bandwidth demand prediction techniques [4] to forecast the user cells' bandwidth demands for the next 10-minute interval. Fig. 4 shows the forecast for users' bandwidth demands of a popular video channel from Beijing in China.

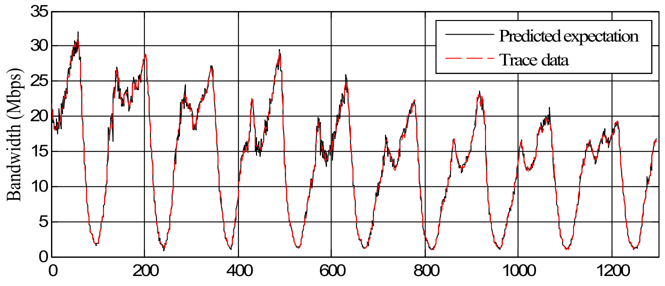


Fig. 4. Ten-minute forecast conditional expectation of users bandwidth demands from Beijing in China for the video channel 142202354.

In this set of experiments, the users are grouped based on their regions (cells) and there are 34 user cells totally. 128 DCs are assumed to be uniformly distributed in these 34 regions. The assumed cloud bandwidth capacity in the first sub-set of the experiments (i.e., 1.3 Gbps per DC) is closer to the peak bandwidth demands of the users. Each DC's bandwidth capacity in the second (i.e., 10 Gbps) and the third (i.e., 25 Gbps) sub-set of experiments are assumed based on the current 10 Gpong and next generation 25 Gpong communication components in the cloud. Using the metrics presented in in Section 5.1, we compare our *DREAM-G*, *DREAM-LG* with *Per-DC Lim* and *Optimal Load Direction* [4]. The latter two algorithms are VoD-specific algorithms and referred to as *PL* and *OD* in this paper respectively. We execute these optimization algorithms every $\delta t = 10$ minutes. The other parameters not mentioned but used in this section are set to the same value as in Section 5.2.

6.2 Performance Observed at VoD Service Providers

Table 7 shows the measures of different algorithms during a 10-day period. Our *DREAM-G* and *DREAM-LG* have lower operating costs, respectively achieving 48.47-66.30 and 75.81-80.17 percent of the cost of the *PL* algorithm, and 53.70-69.40 and 82.66-94.74 percent of the cost of the *OD* algorithm. In other words, our algorithms can save 5.26-51.53 percent total cloud operating costs of *PL* and *OD*.

Because *PL* and *OD* do not consider the optimization of the delivery cost, they incur higher delivery costs than *DREAM-G* and *DREAM-LG*. Our algorithms obtain 51-94 percent 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-LG* means that sometimes users may end up obtaining more expensive bandwidth from local DCs, which explains why the delivery cost of *DREAM-LG* is higher than that of *DREAM-G*. Meanwhile, to provide access locality, *DREAM-LG* must move and store

TABLE 7
Performance of the 4 Algorithms with Different Datacenter Bandwidth Capacities During 10 Days

(a) Each datacenter has 1.3 Gbps bandwidth.

Algorithm	QoS Measures (averaged over all intervals)			System Measures (averaged over all intervals)		Cost Measures (sum over all intervals)			
	Avail	*Bdth*	*Local*	*Rep*	*Crt*	*Dc($)*	*Sc($)*	*Mc($)*	*Oc($)(normalized to PL)*
DREAM-G	100%	99.91%	2.92%	2.94	100.61%	481,263	9.06	1,886.25	483,158(66.30%)
DREAM-LG	100%	99.95%	95.05%	21.91	16.25%	578,763	75.70	5,391.65	584,230(80.17%)
PL	63.61%	99.96%	3.10%	2.55	100%	685,499	8.81	43,231.25	728,739(100%)
OD	100%	99.98%	3.08%	45.72	100.04%	683,408	150.00	12,684.66	696,242(95.54%)

(b) Each datacenter has 10 Gbps bandwidth.

Algorithm	QoS Measures (averaged over all intervals)			System Measures (averaged over all intervals)		Cost Measures (sum over all intervals)			
	Avail	*Bdth*	*Local*	*Rep*	*Crt*	*Dc($)*	*Sc($)*	*Mc($)*	*Oc($)(normalized to PL)*
DREAM-G	100%	99.95%	2.77%	2.59	101.63%	318,083	7.70	189.91	318,280(49.24%)
DREAM-LG	100%	99.95%	95.08%	23.78	0.36%	487,355	79.82	2,525.11	489,960(75.81%)
PL	33.53%	99.96%	2.86%	2.09	100%	611,966	6.78	34,354.86	646,328(100%)
OD	92.33%	99.98%	2.94%	6.40	101.15%	591,063	21.22	1,646.17	592,731(91.71%)

(c) Each datacenter has 25 Gbps bandwidth.

Algorithm	QoS Measures (averaged over all intervals)			System Measures (averaged over all intervals)		Cost Measures (sum over all intervals)			
	Avail	*Bdth*	*Local*	*Rep*	*Crt*	*Dc($)*	*Sc($)*	*Mc($)*	*Oc($)(normalized to PL)*
DREAM-G	100%	99.95%	2.94%	2.57	100.25%	303,745	7.60	41.22	303,794(48.47%)
DREAM-LG	100%	99.95%	95.07%	23.60	0.35%	476,320	79.40	2,422.86	478,822(76.40%)
PL	33.31%	99.97%	2.83%	2.17	100%	590,819	7.08	35,901.09	626,727(100%)
OD	77.25%	99.98%	2.94%	2.86	100.62%	504,843	9.94	580.64	505,434(80.65%)

Avail: *Service availability;* Bdth: *Bandwidth Provision;* Local: *Service access locality;* Rep: *Replication degree;* 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* PL *algorithm.*

more channel replicas to the regions where users' requests are originated, so *DREAM-LG* has higher storage and migration costs than *DREAM-G*. Comparing *DREAM-LG* with *DREAM-G*, the access locality provided by the former incurs 21-58 percent 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-verus-cost tradeoff can often be made at the SLA negotiation time. When compared with *PL* and *OD*, however, *DREAM-LG* offers consistent and impressive operating-cost advantages, as shown in Table 7.

The *PL* algorithm limits the number of channels stored in each DC, so it has lower storage cost, achieving 5.87-71.23 percent of that of the *OD* algorithm. But this limitation on the number of channels per DC severely weakens the service availability of *PL*, as shown in Table 7.

Both *OD* and *PL* 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 *PL* and *OD*. In addition, because many replicas of channels have already been stored in the DCs, *OD* is able to avoid some migration operations of *PL*. On the other hand, when the DCs have larger bandwidth capacities, *OD*, by reserving bandwidth from and placing replicas in a very small subset of DCs, is able to reduce storage cost and migration cost further. In addition, as shown in Table 7, our two algorithms book the minimum necessary bandwidth and achieve comparable bandwidth provision to *PL* and *OD*.

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 element of VoD applications operating cost in the current cloud platform, accounting for at least 99 percent 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.

6.3 QoS of VoD Service

As shown in Table 7, because of integrating availability constraint (Constraint 14) in our model, our algorithms can meet all channels' availability demands completely. Since *DREAM-LG* must replicate channels into many regions to allow users to access data from local DCs, it has a high replication degree. In Table 7, of all the algorithms that meet all availability demands, *DREAM-G* has the lowest storage cost and replication degree (replicas per channel). *PL* and *OD* do not provide any strategy for maintaining availability. But when the DCs have smaller bandwidth capacities, *OD* 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 *OD*, hence the service availability drops to a lower level as shown in Tables 7b and 7c. In all cases, *PL* provides the lowest availability. As shown in Table 7, all four algorithms provide comparable bandwidth provision. Now we investigate the access locality. As shown in Table 7, *DREAM-LG* significantly improves the access locality by integrating locality awareness. This means that *DREAM-LG* utilizes the vast majority of the local bandwidth resources to serve the users so as to reduce the startup latency and cross-region traffic. Because most of the time the bandwidth demands is much less than the peak demand as show in Fig. 4, cloud bandwidth is relatively sufficient. Thus the service locality in the first sub-set of the experiments is closer to that in two other sub-sets of the experiments. 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 7 shows the cross-region traffic generated by different algorithms, normalized to that of the *PL* algorithm. As the *DREAM-G*, *PL* and *OD* algorithms do not have locality awareness, it is thus not surprising to see their cross-region traffic much higher than that of the *DREAM-LG* algorithm.

7 Conclusions

In this paper, we propose a class of distributed holistic algorithms, *DREAM-L*, *DREAM-G* and *DREAM-LG*, to improve the availability of bandwidth-intensive applications with multiple data objects, and provide sufficient DCs' bandwidth to users as well as access locality in a cost-effective way. Extensive experimental evaluation demonstrates significant advantages of our algorithms over the state-of-the-art algorithm *Decision Tree* in providing access locality and meeting users' bandwidth demands. Compared to *Decision Tree*, our *DREAM-LG* improves bandwidth provision and access locality by 11.7-94.1 and 22.9-140.6 percent respectively at a reasonable cloud operating cost. In addition, unlike the majority of the current algorithms solving an optimization problem that suffer from low computational efficiency when the number of data objects is huge, it only takes our *DREAM-LG* algorithm a small time interval to solve the optimization problem owing to the object-grouping technique we propose. In the extensive experiments driven by the traces collected from a large-scale real-world VoD system, we observe that, compared to the two existing state-of-the-art offline algorithms, *Per-DC Lim* and *Optimal Load Direction*, our offline optimization algorithms are able to provide perfect data availability and high access locality and achieve comparable bandwidth provision at only 50-90 percent of the cloud cost of the existing state-of-the-art algorithms.

Acknowledgments

This research is partially supported by the National Natural Science Foundation of China under Grant Nos. 61232004 and 61502189. 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. Ke Zhou is the corresponding author.

References

[1] Ponemon Institute, "2013 Cost of data center outages," USA, 2013.

[2] P. Bodik, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, I. Stoica, "Surviving failures in bandwidth constrained datacenters," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2012, pp. 431–442.

[3] N. Bonvin, T. G. Papaioannou, and K. Aberer, "A self organized, fault-tolerant and scalable replication scheme for cloud storage," in *Proc. 1st ACM Symp. Cloud Comput.*, 2010, pp. 205–216.

[4] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proc. INFOCOM*, 2012, pp. 460–468.

[5] (2015). Amazon Simple Storage Service (S3) [Online]. Available: http://aws.amazon.com/s3/

[6] (2015). Windows Azure [Online]. Available: http://www.windowsazure.com/

[7] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, "FairCloud: Sharing the network in cloud computing," in *Proc. INFOCOM*, 2012, pp. 187–198.

[8] F. Wang, J. Liu, M. Chen, "CALMS: Cloud-assisted live media streaming for globalized demands with time/region diversities," in *Proc. INFOCOM*, 2012, pp. 199–207.

[9] S. J. Liebowitz, "File sharing: Creative destruction or just plain destruction," *The J. Law Econ.*, vol. 49, pp. 1–28, 2005.

[10] (2015). YouTube [Online]. Available: http://www.youtube.com/

[11] U. Sharma, P. Shenoy, S. Sahu and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *Proc. 31st Int. Conf. Distrib. Comput. Syst.*, Jun. 2011, pp. 559–570.

[12] S. Zaman and D. Grosu, "A distributed algorithm for the replica placement problem," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1455–1468, Sep. 2011.

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

[14] E. Pinheiro, W. D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," in *Proc. 5th USENIX Conf. File Storage Technol.*, 2007, p. 2.

[15] R. Miller, "The year in Downtime: The Top 10 outages of 2013," Data Center Knowledge news, USA, 2013, http://www.datacenterknowledge.com/archives/2013/12/16/year-downtime-top-10-outages-2013

[16] R. Miller, "Data center fire leads to outage for samsung devices," Data Center Knowledge news, USA, 2014, http://www.datacenterknowledge.com/archives/2014/04/20/data-center-fire-leads-outage-samsung-devices

[17] (2010). The avoidable cost of downtime. ca Technologies [Online]. Available: http://goo.gl/SC6Ve.

[18] A. Mahimkar, A. Chiu, R. Doverspike, M. D. Feuer, P. Magill, E. Mavrogiorgis, J. Pastor, S. L. Woodward, and J. Yates, "Bandwidth on demand for inter-data center communication," in *Proc. 10th ACM Workshop Hot Topics Netw.*, 2011, pp. 1–6.

[19] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," in *Proc. SIGCOMM*, 2013, pp. 3–14.

[20] N. Carlsson, G. Dan, D. Eager, and A. Mahanti, "Tradeoffs in cloud and peer-assisted content delivery systems," in *Proc. IEEE 12th Int. Conf. Peer-to-Peer Comput.*, 2012, pp. 249–260.

[21] J. Ciancutti, "Four reasons we choose Amazon's cloud as our computing platform," The Netflix " Tech" Blog, 2010, http://techblog.netflix.com/2010/12/four-reasons-we-choose-amazons-cloud-as.html

[22] Y. Wu, C. Wu, B. Li, X. Qiu, F. C. Lau, "CloudMedia: When cloud on demand meets video on demand," in *Proc. 31st Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 268–277.

[23] Datastax Corporation, "Introduction to multi-data center operations with apache cassandra and datastax enterprise," White Paper, 2013.

[24] A. Grama, G. Karypis, V. Kumar, and A. Gupta, *Introduction to Parallel Computing*, 2nd ed. Reading, MA, USA: Addison-Wesley, 2003, ch. 4.

[25] Youku [Online]. Available: http://www.youku.com/

[26] C. Chekuri and S. Khanna, "A polynomial time approximation scheme for the multiple knapsack problem," *SIAM J. Comput.*, vol. 35, no. 3, pp. 713–728, 2005

[27] D. B. Shmoys and E. Tardos, "An approximation algorithm for the generalized assignment problem," *Math. Program.*, vol. 62, no. 3, pp. 461–474, 1993.

[28] C. Chekuri and S. Khanna, "A PTAS for the multiple knapsack problem," in *Proc. 11th Annu. ACM-SIAM Symp. Discr. Algorithms*, 2000, pp. 213–222.

[29] T. Moscibroda and R. Wattenhofer, "Facility location: Distributed approximation," in *Proc. 24th Annu. ACM Symp. Principles Distrib. Comput.*, 2005, pp. 108–117.

[30] D. P. Bertsekas and D. A. Castanon, "The auction algorithm for the transportation problem," *Annu. Oper. Res.*, vol. 20, nos. 1-4, pp. 67–96, 1989.

[31] D. Bertsekas, "A distributed algorithm for the assignment problem," Lab. Inf. Decision Syst. Unpublished Rep., M.I.T., USA, 1979.

[32] T. Loukopoulos and I. Ahmad, "Static and adaptive distributed data replication using genetic algorithms," *J. Parallel Distrib. Comput.*, vol. 64, pp. 1270–1285, 2004.

[33] S. Khan and I. Ahmad, "Heuristic-based replication schemas for fast information retrevial over the Internet," in *Proc. 17th Int. Conf. Parallel Distrib. Comput. Syst.*, 2004, pp. 278–283.

[34] C. Castillo, A. N. Tantawi, D. Arroyo, M. Steinder, "Cost-aware replication for dataflows," in *Proc. IEEE Netw. Oper. Manage. Symp.*, 2012, pp. 171–178.

[35] T. Loukopoulos, I. Ahmad, D. Papadias, "An overview of data replication on the Internet," in *Proc. 6th Int. Symp. Parallel Archit. Algorithm Netw.*, 2002, pp. 27–32.

[36] 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 Int. Conf. Parallel Distrib. Syst.*, 2012, pp. 299–306.

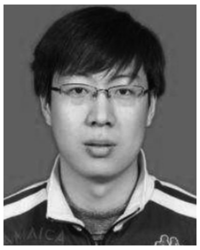
[37] P. Gao, A. Curtis, B. Wong, and S. Keshav, "It's not easy being green," in *Proc. SIGCOMM*, 2012, pp. 211–222.

[38] Y. Zhao, H. Jiang, K. Zhou, Z. Hang, P. Huang, "Meeting service level agreement cost-effectively for video-on-demand applications in the cloud," in *Proc. INFOCOM*, 2014, pp. 298–306.

[39] Q. Wei, B. Veeravalli, B. Gong, L. Zeng, D. Feng, "CDRM: A cost-effective dynamic replication management scheme for cloud storage cluster," in *Proc. IEEE Int. Conf. Cluster Comput.*, 2010, pp. 188–196.

[40] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci, J. Haridas, C. Uddaraju, H. Khatri, A. Edwards, V. Bedekar, S. Mainali, R. Abbasi, A. Agarwal, M. F. Haq, M. I. Haq, D. Bhardwaj, S. Dayanand, A. Adusumilli, M. McNett, S. Sankaran, K. Manivannan, and L. Rigas, "Windows azure storage: A highly available cloud storage service with strong consistency," in *Proc. 23rd ACM Symp. Oper. Syst. Principles*, 2011, pp. 143–157.

[41] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in *Proc. Usenix Annu. Tech. Conf.*, 2012, p. 2.



Yuhong Zhao received the bachelor's and master's degrees in computer science and technology from Heilongjiang University, China, in 2006 and 2010, respectively; and the PhD degree in computer science and technology from the Huazhong University of Science and Technology (HUST), China, in 2015. He is currently an assistant research fellow of the Institute of Information Engineering, Chinese Academy of Sciences. His main research interests include computer architecture, cloud computing, computer storage system, network security, and big data.

Hong Jiang received the BSc degree in computer engineering in 1982 from the Huazhong University of Science and Technology, Wuhan, China; the MASc degree in computer engineering in 1987 from the University of Toronto, Toronto, Canada; and the PhD degree in computer science in 1991 from the Texas A&M University, College Station, TX. He is currently Wendell H. Nedderman Endowed professor and the department chair of computer science and engineering at the University of Texas at Arlington. Prior to joining UTA, he served as a program director at National Science Foundation from January 2013 to August 2015 and he was at the University of Nebraska-Lincoln since 1991, where he was a Willa Cather professor of computer science and engineering. He has graduated 13 PhD students who upon their graduations either landed academic tenure-track positions in PhD-granting US institutions or were employed by major US IT corporations. His present research interests include computer architecture, computer storage systems and parallel I/O, high-performance computing, big data computing, cloud computing, and performance evaluation. He recently served as an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*. He has more than 200 publications in major journals and international Conferences in these areas, including *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *ACM Transactions on Architecture and Code Optimization*, *Journal of Parallel and Distributed Computing*, ISCA, MICRO, USENIX ATC, FAST, LISA, ICDCS, IPDPS, MIDDLEWARE, OOPLAS, ECOOP, SC, ICS, HPDC, INFOCOM, ICPP, etc., and his research has been supported by US National Science Foundation (NSF), DOD, and the State of Nebraska. He is a fellow of the IEEE and a member of ACM.

Ke Zhou received the BE, ME, and PhD degrees in computer science 1and technology from the Huazhong University of Science and Technology (HUST), China, in 1996, 1999, and 2003, respectively. He is a professor of the School of Computer Science and Technology, HUST. His main research interests include computer architecture, cloud storage, parallel I/O and storage security. He has more than 50 publications in journals and international conferences, includ-ing *Performance Evaluation*, FAST, MSST, ACM MM, SYSTOR, MASCOTS, and ICC. He is a member of the IEEE.

Zhijie Huang received the BE degree from Jimei University, Xiamen, in 2005 and the ME degree from the University of Jiangsu, in 2010, both in computer science and technology. He is currently working toward the PhD degree in the School of Computer Science and Technology at the Huazhong University of Science and Technology. His research interests include coding theory, dependable and secure computing, storage systems, and parallel I/O.

Ping Huang received the PhD degree from the HuaZhong University of Science and Technology in 2013. He is currently a postdoctoral researcher fellow in the Department of Electrical and Computer Engineering at Virginia Commonwealth University. His main research interest includes nonvolatile memory, operating system, distributed systems, DRAM, GPU, Key-value systems, etc. He has published papers in various international conferences and journals, including SYSTOR, NAS, MSST, USENIX ATC, Eurosys, Performance, INFOCOM, SRDS, MASCOTS, CCGrid, IPCCC, *Journal of Systems Architecture (JSA)*, and *Performance Evaluation (PEVA)*.