Resource Auto-Scaling and Sparse Content Replication for Video Storage Systems

DI NIU[1], HONG XU[2] and BAOCHUN LI[3], University of Alberta[1], City University of Hong Kong[2], University of Toronto[3]

Many video-on-demand (VoD) providers are relying on public cloud providers for video storage, access and streaming services. In this paper, we investigate how a VoD provider may make optimal bandwidth reservations from a cloud service provider to guarantee the streaming performance while paying for the bandwidth, storage and transfer cost. We propose a predictive resource auto-scaling system that dynamically books the minimum amount of bandwidth resources from multiple servers in a cloud storage system, in order to allow the VoD provider to match its short-term demand projections. We exploit the anti-correlation between the demands of different videos for statistical multiplexing to hedge the risk of under-provisioning. The optimal load direction from video channels to cloud servers without replication constraints is derived with provable performance. We further study the joint load direction and sparse content placement problem that aims to reduce bandwidth reservation cost under sparse content replication requirements. We propose several algorithms, and especially an iterative L_1-norm penalized optimization procedure to efficiently solve the problem while effectively limiting the video migration overhead. The proposed system is backed up by a demand predictor that forecasts the expectation, volatility and correlation of the streaming traffic associated with different videos based on statistical learning. Extensive simulations are conducted to evaluate our proposed algorithms, driven by the real-world workload traces collected from a commercial VoD system.

Categories and Subject Descriptors: Information systems [**Information storage systems**]: Storage architectures—*Cloud based storage*; Networks [**Network performance evaluation**]: —*Network performance modeling*

General Terms: Performance Evaluation; Algorithms; Optimization.

Additional Key Words and Phrases: Video-on-Demand; Cloud computing; Auto-scaling; Content placement; Load direction; Optimization; Sparse design; Prediction.

ACM Reference Format:

ACM Trans. Appl. Percept. 1, 1, Article 1 (January 1111), 30 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Cloud computing is redefining the way many Internet services operate, including Video-on-Demand (VoD). Instead of buying racks of servers and building private datacenters, it is now common for VoD service providers to leverage the computing, network and storage resources of cloud service providers for video storage and streaming. As an example, Netflix places its video data stores, streaming servers,

Author's contact information: D. Niu, email: dniu@ualberta.ca; H. Xu, email: henry.xu@cityu.edu.hk; B. Li, email: bli@ece.toronto.edu. Part of this work has been presented at IEEE INFOCOM 2012.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 1111 ACM 1544-3558/1111/01-ART1 $15.00
DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

encoding software, and other customer-oriented APIs all in Amazon Web Services (AWS) [Netflix 2010].

One of the most important economic appeals of cloud computing is its elasticity and auto-scaling in resource provisioning. Traditionally, after careful capacity planning, an enterprise makes long-term investments on its infrastructure to accommodate its peak workload. Over-provisioning is inevitable while utilization remains low during most non-peak times. In contrast, in the cloud, the number of computing instances launched can be changed adaptively at a fine granularity with a lead time of minutes. This converts the up-front infrastructure investment to operating expenses charged by cloud service providers. As the cloud's auto-scaling ability enhances resource utilization by closely matching supply with demand, the overall expense of the enterprise may be reduced.

Unlike web servers or scientific computing, VoD is a network-bound service with stringent bandwidth requirements. As users must download at a rate no smaller than the video playback rate to smoothly watch video streams online, bandwidth constitutes the performance bottleneck. Thanks to the recent advances in datacenter network virtualization [Bari et al. 2013], bandwidth reservation is likely to become a near-term value-added feature offered by cloud services to appeal to customers with bandwidth-intensive applications like VoD. In fact, there have already been proposals from the perspective of datacenter engineering to offer bandwidth guarantees for egress traffic from a virtual machine (VM), as well as among VM themselves [Guo et al. 2010], [Ballani et al. 2011], [Xie et al. 2012].

In this paper, we analyze the benefits and address open challenges of cloud resource auto-scaling for VoD applications. The benefit of auto-scaling for a video storage and streaming service is intuitive and natural. As shown in Fig. 1(a), traditionally, a VoD provider acquires a monthly plan from ISPs, in which a fixed bandwidth capacity, e.g., 1 Gbps, is guaranteed to accommodate the anticipated peak demand. As a result, resource utilization is low during non-peak times of demand troughs. Alternatively, a pay-as-you-go charge model may be adopted by a cloud provider as shown in Fig. 1(b), where a VoD provider pays for the total amount of bytes transferred. However, the bandwidth capacity available to the VoD provider is subject to variation due to contention from other applications, incurring unpredictable quality-of-service (QoS) issues. Fig. 1(c) illustrates bandwidth auto-scaling and reservation to match demands with appropriate resources, leading to both high resource utilization and QoS guarantees. Apparently, the more frequently the rescaling happens, the more closely resource supply will match the demand.

However, a number of important challenges need to be addressed to achieve auto-scaling in a video storage and streaming service. *First*, since resource rescaling requires a delay of at least a couple of minutes to update configuration and move objects if necessary, it is best to predict the demand with a lead time greater than the update interval, and scale the capacity to meet anticipated demand. Such a proactive, rather than reactive, strategy for resource provisioning needs to consider not only conditional mean demands but also demand fluctuations in order to prevent under-provisioning risks. *Second*, as statistical multiplexing can smooth traffic, a VoD provider may reserve less bandwidth to guard against fluctuations if it *jointly* reserves bandwidth for all its video accesses. However, in a cloud storage system, the content is usually replicated on multiple servers to introduce reliability in the presence of failures and to enable load balancing. The key question is—how should a VoD provider optimally split and direct its workload across the cluster of servers (whether virtual or physical) provided by the cloud service, in order to save the overall bandwidth reservation cost? Furthermore, video content must be replicated across different servers in a sparse way to avoid a high storage cost.

In this paper, we propose a bandwidth auto-scaling facility that dynamically reserves resources from a tightly connected server cluster for VoD providers, with several distinct characteristics. *First*, it is predictive. The facility tracks the history of bandwidth demand for each video using cloud monitor-

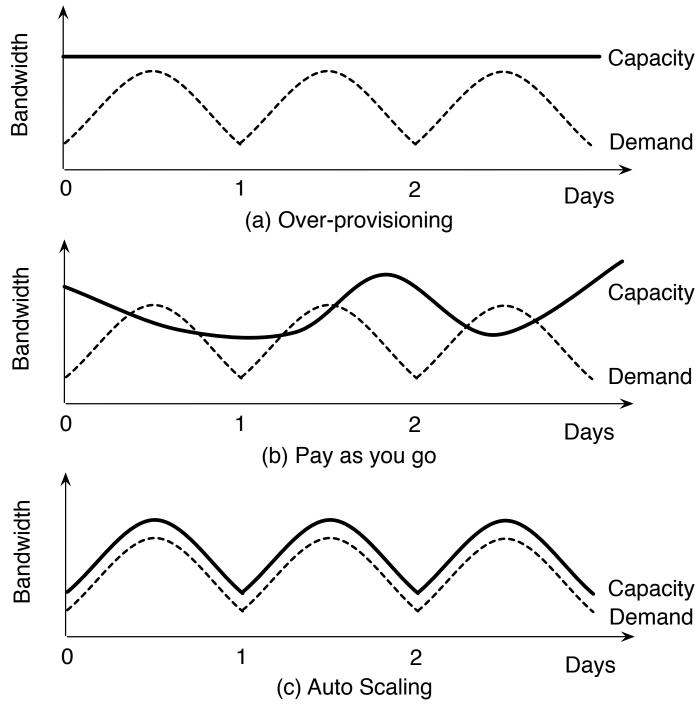


Fig. 1: Bandwidth auto-scaling with quality assurance, as compared to provisioning for the peak demand and pay-as-you-go.

ing services, and periodically estimates the expectation, volatility, and correlations of demands for all videos for the near future using statistical analysis. We propose a novel *video channel interleaving scheme* that can even predict demand for new videos that lack historical demand data. *Second*, it provides QoS assurance by judiciously deciding the minimum bandwidth reservation required to satisfy the demand with high probability. *Third*, it optimally mixes demands based on statistical anti-correlation to save the aggregate bandwidth capacity reserved from all the servers, under the condition that the content must be sparsely replicated with limited content migration.

Given the predicted demand statistics as input, we formulate the bandwidth minimization problem to jointly decide load direction and sparse content placement as a combinatorial problem involving L_0 norms that model content placement sparsity. We derive the theoretically optimal load direction across servers when full replication is permitted, and propose several approximate solutions to the joint load direction and sparse content placement problem, striking a balance between bandwidth and storage costs. In particular, as a highlight, we novelly apply an iteratively reweighted L_1-norm relaxation technique to approximately solve the L_0-norm penalized optimization problem. Our technique not only yields sparse content placement decisions but also effectively reduces the content migration overhead.

We have performed extensive evaluation of the proposed autoscaling strategies for video storage systems, through trace-driven simulations based on the video streaming traces of 1693 video channels collected from UUSee [Liu et al. 2010], a production VoD system, over a 21-day period.

2. SYSTEM ARCHITECTURE

Consider a VoD service provider hosting N videos and relying on S (collocated) servers in a cloud storage system for service. We propose an unobtrusive auto-scaling system that makes predictions about

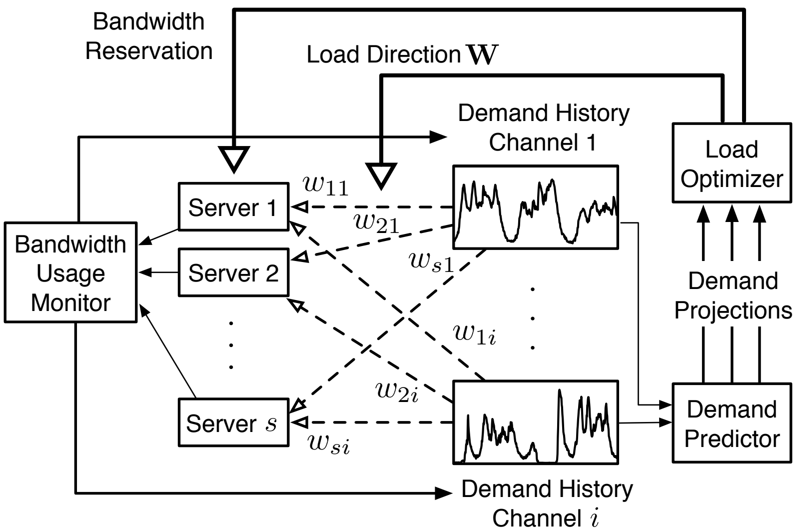


Fig. 2: The system decides the bandwidth reservation from each server and a matrix $\mathbf{W} = [w_{si}]$ every Δt minutes, where w_{si} is the proportion of video channel i's requests directed to server s.

future demands of all videos and reserves minimal necessary resources from the server cluster to satisfy the demand. Our system architecture is shown in Fig. 2, which consists of three key components: *bandwidth usage monitor*, *demand predictor*, and *load optimizer*. Bandwidth rescaling is performed proactively every Δt minutes, with the following three steps:

First, before time t, the system collects bandwidth demand history of all videos up to time t, which can easily be obtained from cloud monitoring services. As an example, Amazon CloudWatch provides a free resource monitoring service to AWS customers at a 5-minute frequency [AWS].

Second, the bandwidth demand history of all videos is fed into the demand predictor to predict the bandwidth requirement of each video for the next Δt minutes, i.e., for the period $[t, t + \Delta t)$. Our predictor not only forecasts the expected demand, but also outputs a *volatility* estimate, which represents the degree that demand will be fluctuating around its expectation, as well as the demand *correlations* between different videos in this period. Our volatility and correlation estimation is based on multivariate GARCH models [Bollerslev 1986], which has gained success in stock analysis and forecast in the past decade.

Finally, the load optimizer takes predicted statistics as the input, calculates the bandwidth capacity to be reserved from each server in the available server pool and determines how many servers should be used. It also outputs a load direction matrix $\mathbf{W} = [w_{si}]$, where w_{si} represents the portion of video i's requests directed to server s. Apparently, we should have $\sum_s w_{si} = 1$ if the aggregate server capacity is sufficient. It is worth noting that the matrix $\mathbf{W}$ also indicates the content placement decision: a copy of video i is placed on server s only if $w_{si} > 0$. In practice, the load direction $\mathbf{W}$ can be readily implemented by routing the requests for video i to server s with probability w_{si}.

The system finishes the above three steps before time t, so that a new bandwidth reservation can be made at time t for the period $[t, t + \Delta t)$. The above process is then repeated for the next period $[t + \Delta t, t + 2\Delta t)$.

Apparently, the key to such a resource autoscaling framework for video storage is the load optimizer, which needs to jointly determine a load direction matrix as well as a sparse content placement strategy to limit both storage and content transfer overhead in each Δt-minute time period. The optimizer

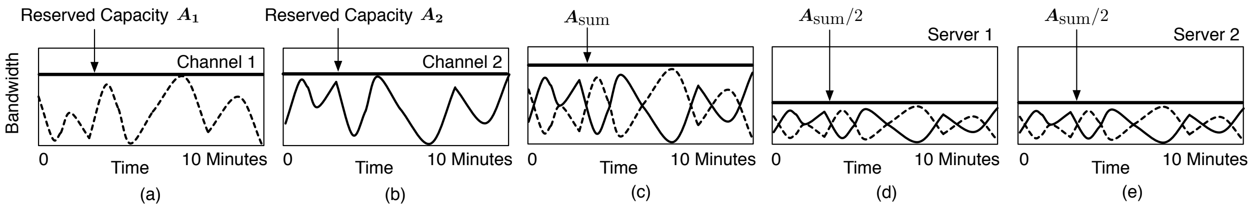


Fig. 3: By exploiting demand correlation between different video channels, we can save the total bandwidth reservation, even within each 10-minute period, while still providing quality assurance to each video channel.

should also determine load direction $\mathbf{W}$ in a way so as to push workloads onto as few servers as possible, which will autoscale the number of servers used.

Bandwidth Reservation vs. Load Balancing. One may be tempted to think that periodic bandwidth reservation is unnecessary, since requests can be flexibly directed to whichever server that has available capacity by a load balancer. However, the latter will exactly fall in the range of pay-as-you-go model with no quality guarantee to VoD users, whereas bandwidth reservation ensures that the provisioned resource can satisfy the projected demand with high probability.

Furthermore, since the content placement is pre-determined in a traditional load balancing system, it is hard to achieve resource autoscaling—it is impossible to push all demands onto as few servers as possible when the total demand shrinks, i.e., the number of servers used is always fixed. Neither can a traditional load balancer adjust content placement dynamically to maximize the multiplexing gain based on demand statistics, as will be discussed subsequently.

Quality-Assured Bandwidth Multiplexing. The bandwidth demand of each video channel can fluctuate drastically even at small time scales. To avoid performance risks, the bandwidth reservation made for each channel in each period should accommodate such fluctuations, inevitably leading to low utilization at troughs, as illustrated in Fig. 3(a) and (b). Trough filling within a short period such as 10 minutes is hard with too many random shocks in demand.

However, our load optimizer strives to enhance utilization even when Δt is as small as 10 minutes by multiplexing demands based on their correlations. The usefulness of anti-correlation is illustrated in Fig. 3(c): if we jointly book capacity for two negatively correlated channels, the total reserved capacity is $A_{\mathsf{sum}} < A_1 + A_2$. Besides aggregation, we can also take a part of demand from each channel, mix them and reserve bandwidth for the mixed demands from multiple servers. As an example, in Fig. 3(d) and (e), the aggregate demand of two channels is split onto two servers, each serving a mixture of demands, which still leads to a total bandwidth reservation of A_{sum}. In each Δt period, we leverage the estimated demand correlations to optimally direct workloads across different servers so that the total bandwidth reservation necessary to guarantee quality is minimized.

Finally, in the case that the actual demand exceeds the reserved bandwidth capacity, the additional requests can still be served in the traditional best-effort fashion.

3. OPTIMAL LOAD DIRECTION AND BANDWIDTH RESERVATION

In this section, we focus on the load optimizer. Suppose before time t, we have obtained the estimates about demands in the upcoming period $[t, t + \Delta t)$. Our objective is to decide load direction $\mathbf{W}$ so as to minimize the total bandwidth reservation while controlling the under-provision risk in each server. The question of how to make demand predictions will be the subject of Sec. 5.

We first introduce a few useful notations. Since we are considering each individual time period, without loss of generality, we drop subscript t in our notations. Recall that the VoD provider runs N video channels. The bandwidth demand of channel i is a random variable D_i with mean μ_i and variance σ_i^2. For convenience, let $\mathbf{D} = [D_1, \ldots, D_N]^\mathsf{T}$, $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_N]^\mathsf{T}$ and $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_N]^\mathsf{T}$.

Note that the random demands $D_1, \ldots, D_N$ may be highly correlated due to the correlation between video genres, viewer preferences and video release times. Denote ρ_{ij} the correlation coefficient of D_i and D_j, with $\rho_{ii} \equiv 1$. Let $\mathbf{\Sigma} = [\sigma_{ij}]$ be the $N \times N$ symmetric demand *covariance matrix*, with $\sigma_{ii} = \sigma_i^2$ and $\sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$ for $i \neq j$.

The VoD provider will book resources from S servers. Denote C_s the upper bound on the bandwidth capacity that can be reserved from server s, for $s = 1, \ldots, S$. C_s may be limited by the available instantaneous outgoing bandwidth at server s, or may be intentionally set by the VoD provider to spread its workload across different servers and avoid booking resources from a single server. Let $C_{\mathsf{sum}} = \sum_s C_s$ be the aggregate utilizable bandwidth capacity of all S servers. Throughout the paper, we assume that C_{sum} is sufficiently large to satisfy all the demands in the system.[1]

Let $\mathbf{\Phi} = [\phi_{si}]$ be the content placement matrix, where $\phi_{si} = 1$ if video i is replicated on server s, and $\phi_{si} = 0$ otherwise. We define a load direction decision as a weight matrix $\mathbf{W} = [w_{si}]$, $s = 1, \ldots, S$, $i = 1, \ldots, N$, where w_{si} represents the portion of video i's demand directed to and served by server s, with $0 \leq w_{si} \leq 1$ and $\sum_s w_{si} = 1$. Apparently, if $\phi_{si} = 0$, we must have $w_{si} = 0$. We observe that $\mathbf{w}_s = [w_{s1}, \ldots, w_{sN}]^\mathsf{T}$ represents the *workload portfolio* of server s. Given $\mathbf{w}_s$, the aggregate bandwidth load imposed on server s is a random variable

$$L_s = \sum_i w_{si} D_i = \mathbf{w}_s^\mathsf{T}\mathbf{D}. \tag{1}$$

We use A_s to denote the amount of bandwidth reserved from server s for this period. Clearly, we must have $A_s \leq C_s$. Let $\mathbf{A} =: [A_1, \ldots, A_S]^\mathsf{T}$. To control the under-provision risk, we require the load imposed on server s to be no more than the reserved bandwidth A_s with high probability, i.e.,

$$\Pr(L_s > A_s) \leq \epsilon, \quad \forall s, \tag{2}$$

where $\epsilon > 0$ is a small constant, referred to as the *under-provision probability*.

3.1 Load Direction under Full Replication

Suppose $\phi_{si} = 1$ for all s, i, i.e., each video is replicated on every server. Then, every w_{si} may take non-zero values. Specifically, given demand expectations $\boldsymbol{\mu}$ and covariances $\mathbf{\Sigma}$, and the available capacities $C_1, \ldots, C_S$, the load optimizer can decide the optimal bandwidth reservation $\mathbf{A}^*$ and load direction $\mathbf{W}^*$ by solving the following optimization problem:

$$\begin{aligned}
\underset{\mathbf{W},\mathbf{A}}{\text{minimize}} \quad & \sum_s A_s && (3)\\
\text{subject to} \quad & A_s \leq C_s, \quad \forall s, && (4)\\
& \Pr(L_s > A_s) \leq \epsilon, \quad \forall s, && (5)\\
& \sum_s w_{si} = 1, \quad \forall i. && (6)
\end{aligned}$$

Through reasonable aggregation, we believe that L_s follows a Gaussian distribution. We will empirically justify this assumption in Sec. 6 using real-world traces. When L_s is Gaussian, constraint (2) is

[1] A rigorous condition for supply exceeding demand is given in Theorem 3.1.

equivalent to

$$A_s \geq \mathbf{E}[L_s] + \theta\sqrt{\mathbf{Var}[L_s]}, \tag{7}$$

with $\theta := F^{-1}(1-\epsilon)$, where $F(\cdot)$ is the CDF of normal distribution $\mathcal{N}(0,1)$. For example, when $\epsilon = 2\%$, we have $\theta = 2.05$. Since

$$\begin{cases} \mathbf{E}[L_s] = \mu_1 w_{s1} + \ldots + \mu_N w_{sN} = \boldsymbol{\mu}^\mathsf{T}\mathbf{w}_s, \\ \mathbf{Var}[L_s] = \sum_{i,j} \rho_{ij}\sigma_i\sigma_j w_{si} w_{sj} = \mathbf{w}_s^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_s, \end{cases}$$

it follows that (2) is equivalent to

$$A_s \geq \boldsymbol{\mu}^\mathsf{T}\mathbf{w}_s + \theta\sqrt{\mathbf{w}_s^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_s}. \tag{8}$$

Therefore, the bandwidth minimization problem (3) under full replication is equivalent to

$$\underset{\mathbf{W}}{\text{minimize}} \quad \sum_s A_s \tag{9}$$

$$\text{subject to} \quad A_s = \boldsymbol{\mu}^\mathsf{T}\mathbf{w}_s + \theta\sqrt{\mathbf{w}_s^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_s}, \tag{10}$$

$$\boldsymbol{\mu}^\mathsf{T}\mathbf{w}_s + \theta\sqrt{\mathbf{w}_s^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_s} \leq C_s, \quad \forall s, \tag{11}$$

$$\sum_s \mathbf{w}_s = \mathbf{1}, \tag{12}$$

$$\mathbf{0} \preceq \mathbf{w}_s \preceq \mathbf{1}, \quad \forall s, \tag{13}$$

where $\mathbf{1} = [1,\ldots,1]^\mathsf{T}$ and $\mathbf{0} = [0,\ldots,0]^\mathsf{T}$ are N-dimensional column vectors.

Under full replication, we can derive nearly closed-form solutions to problem (9) in the following theorem:

THEOREM 3.1. *If $C_{\mathsf{sum}} \geq \boldsymbol{\mu}^\mathsf{T}\mathbf{1} + \theta\sqrt{\mathbf{1}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{1}}$, an optimal load direction matrix $[w^*_{si}]$ is given by*

$$w^*_{si} = \alpha_s, \quad \forall i, \quad s = 1,\ldots,S, \tag{14}$$

where $\alpha_1,\ldots,\alpha_S$ can be any solution to

$$\begin{cases} \sum_s \alpha_s = 1, \\ 0 \leq \alpha_s \leq \min\left\{1, \frac{C_s}{\boldsymbol{\mu}^\mathsf{T}\mathbf{1}+\theta\sqrt{\mathbf{1}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{1}}}\right\}, \ \forall s. \end{cases} \tag{15}$$

If $C_{\mathsf{sum}} < \boldsymbol{\mu}^\mathsf{T}\mathbf{1} + \theta\sqrt{\mathbf{1}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{1}}$, there is no feasible solution that satisfies constraints (11) to (13).

Proof Sketch: First, $f(\mathbf{w}_s) = \sqrt{\mathbf{w}_s^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_s}$ is a cone and thus a convex function. Hence, we have

$$f\left(\frac{\mathbf{w}_1 + \mathbf{w}_2}{2}\right) \leq \frac{f(\mathbf{w}_1) + f(\mathbf{w}_2)}{2},$$

or equivalently,

$$\sqrt{(\mathbf{w}_1 + \mathbf{w}_2)^\mathsf{T}\boldsymbol{\Sigma}(\mathbf{w}_1 + \mathbf{w}_2)} \leq \sqrt{\mathbf{w}_1^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_1} + \sqrt{\mathbf{w}_2^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_2}.$$

By induction, we can prove

$$\sum_s \sqrt{\mathbf{w}_s^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_s} \geq \sqrt{\Big(\sum_s \mathbf{w}_s^\mathsf{T}\Big)\boldsymbol{\Sigma}\Big(\sum_s \mathbf{w}_s\Big)}. \tag{16}$$

If $\sum_s \mathbf{w}_s = \mathbf{1}$ is feasible, by (11) and (16) we have

$$\begin{aligned}\sum_s C_s &\geq \boldsymbol{\mu}^\top \mathbf{1} + \theta\sqrt{\Big(\sum_s \mathbf{w}_s^\top\Big)\boldsymbol{\Sigma}\Big(\sum_s \mathbf{w}_s\Big)} \\ &= \boldsymbol{\mu}^\top \mathbf{1} + \theta\sqrt{\mathbf{1}^\top\boldsymbol{\Sigma}\mathbf{1}}.\end{aligned}$$

If $\sum_s C_s \geq \boldsymbol{\mu}^\top \mathbf{1} + \theta\sqrt{\mathbf{1}^\top\boldsymbol{\Sigma}\mathbf{1}}$, it is easy to verify (15) is feasible. When $w_{si}^* = \alpha_s$ given by (14), we find (11), (12) and (13) are all satisfied. Hence, (14) is a feasible solution and $\sum_s \mathbf{w}_s = \mathbf{1}$ is feasible. By (16), the objective (9) satisfies

$$\begin{aligned}&\sum_s \Big(\boldsymbol{\mu}^\top \mathbf{w}_s + \theta\sqrt{\mathbf{w}_s^\top\boldsymbol{\Sigma}\mathbf{w}_s}\Big) \\ \geq&\boldsymbol{\mu}^\top \sum_s \mathbf{w}_s + \theta\sqrt{\Big(\sum_s \mathbf{w}_s^\top\Big)\boldsymbol{\Sigma}\Big(\sum_s \mathbf{w}_s\Big)} \\ =&\boldsymbol{\mu}^\top \mathbf{1} + \theta\sqrt{\mathbf{1}^\top\boldsymbol{\Sigma}\mathbf{1}}.\end{aligned} \tag{17}$$

We find that $[w_{si}^*]$ given by (14) achieves the above inequality with equality, and thus is also an optimal solution to (9). □

Theorem 3.1 implies that in the optimal solution, each video channel should split and direct its workload to all S servers following the same weights $\alpha_1, \ldots, \alpha_S$, which can be found by solving the linear constraints (15). Moreover, the optimal workload portfolio of each server s has a similar structure of $\mathbf{w}_s = \alpha_s\mathbf{1}$, where α_s depends on its available capacity C_s through the constraints (15).

Under the optimal load direction, the aggregate bandwidth reservation reaches its minimum value:

$$\begin{aligned}\sum_s A_s^* &= \sum_s \Big(\boldsymbol{\mu}^\top \mathbf{w}_s^* + \theta\sqrt{\mathbf{w}_s^{*\top}\boldsymbol{\Sigma}\mathbf{w}_s^*}\Big) \\ &= \boldsymbol{\mu}^\top \mathbf{1} + \theta\sqrt{\mathbf{1}^\top\boldsymbol{\Sigma}\mathbf{1}},\end{aligned}$$

which does not depend on S, the number of servers. This means that having demand served by multiple servers instead of one big server does not increase bandwidth reservation cost as long as $w_{si} = \alpha_s$, $\forall i$ given by (14). Therefore, the load optimizer can first aggregate all the demands and then split the aggregated demand into different servers subject to their capacities.

3.2 Load Direction under Limited Replication

Although solution (14) is optimal in terms of bandwidth reservation, it encounters two major obstacles in practice. *First*, as long as $\alpha_s > 0$, $w_{si}^* = \alpha_s > 0$ for all i, which means that server s has to store all N videos. In other words, a video has to be replicated on to every server s that has $\alpha_s > 0$. This incurs significant additional storage cost. *Second*, each video channel i splits its workload onto all S servers according to the weights $\alpha_1, \ldots, \alpha_S$. When S is large and D_i is small, such fine-grained splitting will not be feasible from an engineering perspective.

Therefore, in practice, each video should only be replicated on a few servers to maintain a reasonable storage overhead. Thus, for each video i, we should have $\phi_{si} = 1$ only for a subset of all s. If the content placement matrix Φ is given, the optimal load direction problem becomes

$$\underset{\mathbf{W}}{\text{minimize}} \quad \sum_s A_s \tag{18}$$

$$\text{subject to} \quad A_s = \boldsymbol{\mu}^\mathsf{T}\mathbf{w}_s + \theta\sqrt{\mathbf{w}_s^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_s}, \tag{19}$$

$$\boldsymbol{\mu}^\mathsf{T}\mathbf{w}_s + \theta\sqrt{\mathbf{w}_s^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_s} \leq C_s, \quad \forall s, \tag{20}$$

$$\sum_s \phi_{si} w_{si} = 1, \quad \forall i, \tag{21}$$

$$\sum_s (1-\phi_{si}) w_{si} = 0, \quad \forall i, \tag{22}$$

$$\mathbf{0} \preceq \mathbf{w}_s \preceq \mathbf{1}, \quad \forall s, \tag{23}$$

As compared to the load direction optimization problem (9) under full replication, problem (18) has the new constraints (21) and (22) due to limited replication, which are clearly equivalent to

$$\begin{cases} \sum_s w_{si} = 1, \ \forall i, \\ w_{si} = 0, \quad , \ \text{if } \phi_{si} = 0 \end{cases}$$

Problem (18) is a convex problem, and in particular, a second-order cone program (SOCP) that can be solved efficiently using standard convex optimization solvers such as interior-point algorithms or active-set methods. Apparently, the minimum achievable total bandwidth reservation $\sum_s A_s^*$ is lower-bounded by the $\sum_s A_s^*$ under full replication, which is $\boldsymbol{\mu}^\mathsf{T}\mathbf{1} + \theta\sqrt{\mathbf{1}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{1}}$.

4. SPARSE CONTENT PLACEMENT DESIGN

In this section, we consider the joint design of content placement matrix Φ and load direction matrix $\mathbf{W}$, given the workload statistics μ and Σ. As has been mentioned in Sec. 2, given a $\mathbf{W}$, Φ can be determined in the following way:

$$\phi_{si}(w_{si}) = \begin{cases} 1, \ \text{if } w_{si} > 0, \\ 0, \ \text{if } w_{si} = 0, \end{cases} \tag{24}$$

which means that video i needs to be replicated on to server s *only if* $w_{si} > 0$. Therefore, to determine load direction with a limited replication overhead, we can use $\mathbf{W}$ as a single decision variable and constrain the number of non-zero entries in it when minimizing bandwidth reservation, leading to the following optimization problem:

$$\underset{\mathbf{W}}{\text{minimize}} \quad \sum_s A_s \tag{25}$$

$$\text{subject to} \quad A_s = \boldsymbol{\mu}^\mathsf{T}\mathbf{w}_s + \theta\sqrt{\mathbf{w}_s^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_s}, \tag{26}$$

$$\boldsymbol{\mu}^\mathsf{T}\mathbf{w}_s + \theta\sqrt{\mathbf{w}_s^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_s} \leq C_s, \quad \forall s, \tag{27}$$

$$\sum_s \mathbf{w}_s = \mathbf{1}, \tag{28}$$

$$\mathbf{0} \preceq \mathbf{w}_s \preceq \mathbf{1}, \quad \forall s, \tag{29}$$

$$\|\mathbf{w}_s\|_0 \leq k_s, \quad \forall s, \tag{30}$$

where $\|\mathbf{w}_s\|_0$ is the l_0 norm of $\mathbf{w}_s$, which represents the number of non-zero components in $\mathbf{w}_s$. Constraint (30) essentially says that each server s should only store up to k_s videos. Apparently, problem 25 involves L_0 norms and is non-convex.

4.1 Per-Server Heuristics

We propose a suboptimal solution to problem (25) that can handle the storage overhead constraint. First, we present a heuristic outlined in Algorithm 1, which outputs $\mathbf{w}_1^{**}, \ldots, \mathbf{w}_S^{**}$ for each server s one after another.

ALGORITHM 1: Per-Server Optimal.

$\mathbf{b} \leftarrow \mathbf{1}$
for $s = 1, \ldots, S$ **do**
  Solve the following problem to obtain $\mathbf{w}_s^{**}$:

$$\underset{\mathbf{w}_s}{\text{maximize}} \quad \boldsymbol{\mu}^\mathsf{T} \mathbf{w}_s \tag{31}$$

$$\text{subject to} \quad \boldsymbol{\mu}^\mathsf{T} \mathbf{w}_s + \theta \sqrt{\mathbf{w}_s^\mathsf{T} \boldsymbol{\Sigma} \mathbf{w}_s} \leq A_s \leq C_s, \tag{32}$$

$$0 \leq \mathbf{w}_s \leq \mathbf{b} \tag{33}$$

  $\mathbf{b} \leftarrow \mathbf{b} - \mathbf{w}_s^{**}$.
  Exit if $\mathbf{b} \leq \mathbf{0}$.
end

Algorithm 1 packs the random demands into each server, one after another, by maximizing the expected demand $\boldsymbol{\mu}^\mathsf{T}\mathbf{w}_s$ each server s can accommodate subject to the probabilistic performance guarantee in (32). As a result, the total amount of resources needed to guard against demand variability is reduced. Clearly, with Algorithm 1, the aggregate bandwidth reservation from all servers is

$$\sum_s A_s^{**} = \sum_{s=1}^{S} (\boldsymbol{\mu}^\mathsf{T} \mathbf{w}_s^{**} + \theta \sqrt{\mathbf{w}_s^{**\mathsf{T}} \boldsymbol{\Sigma} \mathbf{w}_s^{**}}). \tag{34}$$

Note that Algorithm 1 is also computationally efficient since (31) is a standard second-order cone program.

Now we handle the constraint $\|\mathbf{w}_s\|_0 \leq k_s$, which requires each server s to store at most k_s videos. We modify Algorithm 1 to cope with this constraint, leading to Algorithm 2, which outputs $\mathbf{w}'_1, \ldots, \mathbf{w}'_S$ for each server s one after another.

ALGORITHM 2: Per-Server Limited Channels.

$\mathbf{b} \leftarrow \mathbf{1}$
for $s = 1, \ldots, S$ **do**
  Solve problem (31) to obtain $\mathbf{w}_s^{**}$
  Choose the top k_s channels with the largest weights and solve problem (31) again only for these k_s channels to obtain $\mathbf{w}'_s$
  $\mathbf{b} \leftarrow \mathbf{b} - \mathbf{w}'_s$.
  Exit if $\mathbf{b} \leq \mathbf{0}$.
end

With Algorithm 2, the aggregate bandwidth reserved is

$$\sum_{s} A'_s = \sum_{s=1}^{S} (\boldsymbol{\mu}^{\mathsf{T}} \mathbf{w}'_s + \theta \sqrt{\mathbf{w}'^{\mathsf{T}}_s \boldsymbol{\Sigma} \mathbf{w}'_s}). \quad (35)$$

In Sec. 6, we will show through trace-driven simulations that Algorithm 2, though suboptimal, effectively limits the content replication degree, thus balancing the savings on storage cost and bandwidth reservation cost for VoD providers.

4.2 Relaxation through Iteratively Weighted L_1 Norms

We now propose to use another algorithm based on L_1-norm relaxation to solve (25) iteratively. Our idea is to adapt a so-called *Log-det* heuristic in sparse recovery to our sparse design problem. The Log-det heuristic has previously been applied to *cardinality minimization* [Fazel et al. 2003], i.e., finding a vector $x = (x_1, \ldots, x_n)$ with the minimum cardinality in a convex set $\mathcal{C}$, or equivalently, minimizing $\|x\|_0 = \sum_i \phi(x_i)$ subject to $x \in \mathcal{C}$, where $\phi(x_i) = 0$ if $x_i = 0$ and $\phi(x_i) = 1$ if $x_i > 0$. The basic idea is to replace the 0-1 valued objective $\phi(x_i)$ for each x_i by a smooth $\log$ function $\log(|x_i|+\delta)$ and to minimize a linearization of $\log(|x_i| + \delta)$ iteratively, which leads to iteratively reweighted L_1-norm approximations to L_0 norms. In sparse recovery, it is shown that such iteratively reweighted L_1 norms can yield more accurate recovery results than one-time L_1 norm approximation [Candes et al. 2008].

To adapt the iteratively reweighted L_1-norm approximation to our sparse design problem, in each iteration, we use a carefully designed convex constraint to replace the L_0-norm constraint (30) and solve the modified problem (25). As iterations proceed, the designed convex constraint is expected to approach the L_0-norm constraint (30) eventually. The algorithm is described in Algorithm 3.

ALGORITHM 3: Iterative L_1-Constrained.

Initially, replace (30) by $\sum_i w_{si} \leq k_s$, for all s, and solve the modified problem (25) under the new constraints to obtain $\mathbf{W}^0$.

for $t = 1, \ldots,$ *maximum iteration* **do**

Given the solution $\mathbf{W}^{t-1}$ in the previous iteration, define $\hat{\phi}^t_{si}$ as

$$\hat{\phi}^t_{si}(w_{si}) = \frac{w_{si}}{w^{t-1}_{si} + \delta}, \quad \forall\, s,\; i \quad (36)$$

Solve the following modified problem to obtain $\mathbf{W}^t$:

$$\begin{aligned}
\underset{\mathbf{W}}{\text{minimize}} \quad & \sum_{s} A_s & (37)\\
\text{subject to} \quad & A_s = \boldsymbol{\mu}^{\mathsf{T}} \mathbf{w}_s + \theta \sqrt{\mathbf{w}_s^{\mathsf{T}} \boldsymbol{\Sigma} \mathbf{w}_s}, & (38)\\
& \boldsymbol{\mu}^{\mathsf{T}} \mathbf{w}_s + \theta \sqrt{\mathbf{w}_s^{\mathsf{T}} \boldsymbol{\Sigma} \mathbf{w}_s} \leq C_s,\ \forall s, & (39)\\
& \sum_{s} \mathbf{w}_s = \mathbf{1}, & (40)\\
& \mathbf{0} \preceq \mathbf{w}_s \preceq \mathbf{1}, \quad \forall s, & (41)\\
& \sum_{i} \hat{\phi}^t_{si}(w_{si}) \leq k_s, \quad \forall s. & (42)
\end{aligned}$$

Break if $\mathbf{W}^t$ and $\mathbf{W}^{t-1}$ are approximately equal.

end

Return $\mathbf{W}^* \leftarrow \mathbf{W}^t$.

Let us explain the rationale behind Algorithm 3. Initially, we replace the constraint $\|\mathbf{w}_s\|_0 \le k_s$ with $\sum_i w_{si} \le k_s$, which is a standard L_1-norm relaxation, since $\|\mathbf{w}_s\|_1 = \sum_i |w_{si}| = \sum_i w_{si}$. It is not hard to see that the bandwidth reservation achieved by $\mathbf{W}^0$ is a lower bound on the optimal value of the original problem (25). The reason is that we have

$$\phi_{si}(w_{si}) \ge w_{si}, \quad \text{if } 0 \le w_{si} \le 1,$$

and thus

$$\|\mathbf{w}_s\|_0 = \sum_i \phi_{si}(w_{si}) \ge \sum_i w_{si}.$$

Therefore, $\sum_i w_{si} \le k_s$ forms a larger region than $\|\mathbf{w}_s\|_0 \le k_s$, and the optimal value achieved by $\mathbf{W}^0$ in the relaxed (convex) problem is a lower-bound of the original optimal value.

Subsequently, in each iteration, the constraint $\|\mathbf{w}_s\|_0 \le k_s$ is replaced by an inequality involving a weighted sum, i.e.,

$$\sum_i \frac{w_{si}}{w_{si}^{t-1} + \delta} \le k_s,$$

which is a generalized version of L_1-norm relaxation with a different weight for each variable. Note that for a sufficiently small δ, upon convergence, i.e., when $w_{si}^{t-1} = w_{si}^t = w_{si}^*$, we have

$$\hat{\phi}_{si}(w_{si}^*) = \frac{w_{si}^*}{w_{si}^* + \delta} \approx \begin{cases} 0 & \text{if } w_{si}^* = 0, \\ 1 & \text{if } w_{si}^* > 0, \end{cases}$$

which is approximately $\phi_{si}(w_{si}^*)$. Thus, the modified constraint $\sum_i \hat{\phi}_{si}^t(w_{si}) \le k_s$ eventually approaches the L_0-norm constraint $\|\mathbf{w}_s\|_0 \le k_s$ in the original problem, and the generated $\mathbf{W}^t$ will almost be feasible for the original problem (25).

4.3 Reducing Migration Overhead and Iterative L_1-Penalized Optimization

A common issue faced by the above schemes is the content migration overhead. As sparse content placement optimization is performed every Δt minutes under varying (predicted) demands, the placement solutions may change from time to time, leading to the overhead of transferring video copies. To mitigate migration overhead, we further propose the following Iterative L_1-Penalized Optimization, which not only yields a sparse placement solution, but also limits the transfer or creation of video copies in each time period by attempting to generate a placement solution that is similar to that of the preceding time period.

First, we add a regularizing term to the original content placement problem (25) to yield

$$\underset{\mathbf{W}}{\text{minimize}} \quad \sum_s A_s + \lambda \sum_{(s,i):w_{si}^{\text{pre}}=0} \phi_{si}(w_{si}) \tag{43}$$

$$\text{subject to} \quad A_s = \boldsymbol{\mu}^\mathsf{T}\mathbf{w}_s + \theta\sqrt{\mathbf{w}_s^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_s}, \tag{44}$$

$$\boldsymbol{\mu}^\mathsf{T}\mathbf{w}_s + \theta\sqrt{\mathbf{w}_s^\mathsf{T}\boldsymbol{\Sigma}\mathbf{w}_s} \le C_s, \quad \forall s, \tag{45}$$

$$\sum_s \mathbf{w}_s = \mathbf{1}, \tag{46}$$

$$\mathbf{0} \preceq \mathbf{w}_s \preceq \mathbf{1}, \quad \forall s, \tag{47}$$

$$\tag{48}$$

where $\lambda > 0$, and w_{si}^{pre} denotes the sparse solution for the previous time period. The regularizer $\sum_{(s,i):w_{si}^{\text{pre}}=0} \phi_{si}(w_{si})$ represents the number of video copies that need to be copied or transferred in this time period; a copy of video i needs to be transferred to server s only if server s doesn't store video i previously, i.e., $w_{si}^{\mathsf{pre}} = 0$, and $w_{si} > 0$ ($\phi_{si}(w_{si}) = 1$) as a result of the current optimization. Note that in this case, we may remove the constraint $\|\mathbf{w}_s\|_0 \le k_s$, since the regularizer itself can already generate sparse solutions, as will be explained subsequently.

To solve problem (43), we apply Algorithm 1 to (43) with iteratively reweighted L_1-norm relaxation for the regularizer, leading to the following Iterative L_1-Penalized algorithm.

ALGORITHM 4: Iterative L_1-Penalized.

$\mathbf{b} \leftarrow \mathbf{1}$
for $s = 1, \ldots, S$ **do**
  Solve the following subproblem using Algorithm 5 to obtain $\mathbf{w}_s^{**}$:

$$\underset{\mathbf{w}_s}{\text{maximize}} \quad \boldsymbol{\mu}^{\mathsf{T}}\mathbf{w}_s - \lambda \sum_{i:w_{si}^{\mathsf{pre}}=0} \phi_{si}(w_{si}) \tag{49}$$

$$\text{subject to} \quad \boldsymbol{\mu}^{\mathsf{T}}\mathbf{w}_s + \theta\sqrt{\mathbf{w}_s^{\mathsf{T}}\boldsymbol{\Sigma}\mathbf{w}_s} \le A_s \le C_s, \tag{50}$$

$$0 \le \mathbf{w}_s \le \mathbf{b} \tag{51}$$

  $\mathbf{b} \leftarrow \mathbf{b} - \mathbf{w}_s^{**}$.
  Exit if $\mathbf{b} \le \mathbf{0}$.
end

ALGORITHM 5: Subroutine to solve the penalized problem (49) in Algorithm 4 for a particular server s.

Initially, $w_{si}^0 = 1 - \delta$, for all i.
for $t = 1, \ldots,$ *maximum iteration* **do**
  Given the solution $\mathbf{w}_s^{t-1}$ in the previous iteration $t-1$, define $\hat{\phi}_{si}^t$ as

$$\hat{\phi}_{si}^t(w_{si}) = \frac{w_{si}}{w_{si}^{t-1} + \delta}, \quad \forall\, i. \tag{52}$$

  Solve the following modified problem to obtain $\mathbf{w}_s^t$:

$$\underset{\mathbf{w}_s}{\text{maximize}} \quad \boldsymbol{\mu}^{\mathsf{T}}\mathbf{w}_s - \lambda \sum_{i:w_{si}^{\mathsf{pre}}=0} \hat{\phi}_{si}^t(w_{si}) \tag{53}$$

$$\text{subject to} \quad \boldsymbol{\mu}^{\mathsf{T}}\mathbf{w}_s + \theta\sqrt{\mathbf{w}_s^{\mathsf{T}}\boldsymbol{\Sigma}\mathbf{w}_s} \le C_s, \tag{54}$$

$$0 \le \mathbf{w}_s \le \mathbf{b} \tag{55}$$

  Break if $\mathbf{w}_s^t$ and $\mathbf{w}_s^{t-1}$ are approximately equal.
end
Return $\mathbf{w}_s^* \leftarrow \mathbf{w}_s^t$.

Now we explain the rationale behind Algorithm 4. Initially, for the first Δt-minute time period, we set $w_{si}^{\mathsf{pre}} = 0$ for all s and i. By doing so, the regularizer in problem (43) will drive the solutions $\mathbf{W}$ to be sparse, such that the placement is sparse. Therefore, by properly setting λ in the regularizer, we do not need to include the constraint $\|\mathbf{w}_s\|_0 \le k_s$. For each Δt-minute time period afterwards, the optimization problem (43) is approximately solved by Algorithm 4 with penalty on the new 1's introduced in matrix $\mathbf{W}$. In other words, if $w_{si}^{\mathsf{pre}} = 0$ in the content placement decision for the previous time period, Algorithm 4 will penalize $w_{si}^{\mathsf{pre}} = 1$ for the current time period, thus preventing new video copies from being created or transferred. Since the content placement for the first time period is sparse and each subsequent period penalizes the difference from the previous period, the content placements for subsequent periods also tend to be sparse. This fact will be demonstrated in Sec. 6 through trace-driven simulations.

5. DEMAND PREDICTION

The derivation of load direction decisions critically depends on parameters $\boldsymbol{u}$ and $\boldsymbol{\Sigma}$, which are estimates of the expected demands and demand covariances for the short-term future $[t, t + \Delta t)$. In this section, we present efficient time series forecasting methods to make such predictions based on past observations.

We assume that the bandwidth demand of channel i at any point in the period $[t, t + \Delta t)$ can be represented by the same random variable D_{it}. This is a reasonable assumption when Δt is small. Similarly, let $\boldsymbol{\mu}_t = [\mu_{1t}, \ldots, \mu_{Nt}]$ and $\boldsymbol{\Sigma}_t = [\sigma_{ijt}]$ represent the demand expectation vector and demand covariance matrix for all N channels in $[t, t+\Delta t)$. We assume that before time t, the system has already collected enough demand history from cloud monitoring services with a sampling interval of Δt. The question is how to use the available sampled bandwidth demand history $\{D_{i\tau} : \tau = 0, \ldots, t-1, i = 1, \ldots, N\}$ to estimate $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$?

In this paper, we combine our previously proposed seasonal ARIMA model [Niu et al. 2011b] for conditional mean (expectation conditioned on the history) prediction with the GARCH model [Niu et al. 2011a] for conditional variance prediction to obtain a *multivariate GARCH* model that can forecast the demand covariance matrix. The model extracts the periodic evolution pattern from each channel's demand time series, and characterizes the remaining *innovation* series as autocorrelated GARCH processes. We briefly describe these statistical models here.

The difficulty in modeling the bandwidth demand of a channel i is that it exhibits diurnal periodicity, a downward trend as the video becomes less popular over time, and changing levels of fluctuation as population goes up and down. Such *non-stationarity* in traffic renders unbiased linear predictors useless. We tackle this problem by applying one-day-lagged differences (the lag is 144 if $\Delta t = 10$ minutes) onto $\{D_{i\tau}\}$ to remove daily periodicity to obtain the transformed series $\{D'_{i\tau} := D_{i\tau} - D_{i\tau-144}\}$, which can be modeled as a low-order autoregressive moving-average (ARMA) process:

$$\begin{cases} D'_{i\tau} - \phi_i D'_{i\tau-1} = N_{i\tau} + \gamma_i N_{i\tau-1}, \\ D'_{i\tau} = D_{i\tau} - D_{i,\tau-144}, \end{cases} \tag{56}$$

where $\{N_{i\tau}\} \sim \mathrm{WN}(0, \sigma^2)$ denotes the uncorrelated white noise with zero mean. Model (56) falls in the category of seasonal ARIMA models [Niu et al. 2011b], [Box et al. 2008].

Model parameters ϕ_i and γ_i in (56) can be trained based on historical data using a maximum likelihood estimator [Box et al. 2008]. To predict the expected demand μ_{it} of channel i, we first predict $\mu'_{it} := \mathbf{E}[D'_{it} | D'_{it-1}, D'_{it-2}, \ldots]$ for the transformed series $\{D'_{i\tau}\}$ to obtain the estimate $\hat{\mu}'_{it}$, using an unbiased *minimum mean square error* (MMSE) predictor. We then retransform $\hat{\mu}'_{it}$ into an estimate $\hat{\mu}_{it}$ of the conditional mean μ_{it}, with the inverse of one-day-lagged differencing.

Given the conditional means $\{\hat{\mu}_{i\tau}\}$ of channel i over all time τ, we denote the *innovations* in $\{D_{i\tau}\}$ by $\{Z_{i\tau}\}$, where

$$Z_{i\tau} := D_{i\tau} - \hat{\mu}_{i\tau}. \tag{57}$$

Since the innovation term $Z_{i\tau}$ represents the fluctuation of $D_{i\tau}$ relative to its projected expectation $\hat{\mu}_{i\tau}$, and such fluctuation may be changing over time, we model the innovations $\{Z_{i\tau}\}$ using a GARCH process:

$$\begin{cases} Z_{i\tau} = \sqrt{h_{i\tau}} e_\tau, \quad \{e_\tau\} \sim IID\mathcal{N}(0,1), \\ h_{i\tau} = \alpha_{i0} + \alpha_{i1} Z_{i\tau-1}^2 + \beta_i h_{i\tau-1}, \end{cases} \tag{58}$$

where $\{Z_{i\tau}\}$ is modeled as a zero-mean Gaussian process yet with a time-varying conditional variance $h_{i\tau}$. Instead of assuming a constant variance for $\{Z_{i\tau}\}$, (58) introduces autocorrelation into volatility evolution and forecasts the conditional variance h_{it} of Z_{it} as a regression of past $h_{i\tau}$ and $Z_{i\tau}^2$. The model parameters in (58) can be learned using maximum likelihood estimation (pp. 417, [Box et al. 2008]) based on training data.

Furthermore, the instantaneous shocks to demands for different videos can be correlated in a large-scale system. An increase in one video's demand may or may not affect the demand for other videos depending on factors like video genres, release time, etc. To incorporate demand correlation, instead of estimating volatility for each video separately, we can estimate the time-varying conditional covariance matrix $\mathbf{\Sigma}_t$ using multivariate GARCH [Enders 2010]. However, multivariate GARCH models are very difficult to estimate for large-scale problems. For the 2-video case, the number of model parameters to estimate in GARCH$(1,1)$ is 21, and for the 3-video case, such a number escalates to 78.

To efficiently predict the covariance matrix $\mathbf{\Sigma}_t$, we introduce a *constant conditional correlation* (CCC) model [Enders 2010], which is a popular multivariate GARCH specification that restricts the correlation coefficients ρ_{ij} to be constant. ρ_{ij} can be estimated as the correlation coefficient between series $\{Z_{i\tau}\}$ and $\{Z_{j\tau}\}$ in recent time periods, and $\rho_{ij} = 1$ if $i = j$. The covariance σ_{ijt} between video i and j at time t is thus predicted as

$$\hat{\sigma}_{ijt} = h_{ijt} = \rho_{ij} \sqrt{h_{it} h_{jt}}, \tag{59}$$

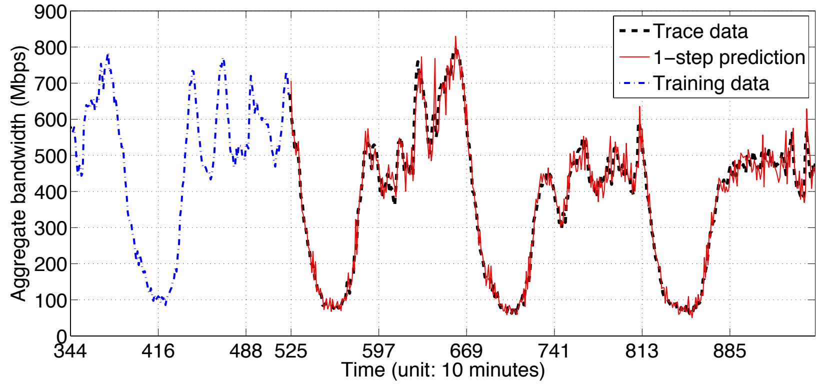
with h_{it} and h_{jt} predicted using (58) for channels i and j individually.

The full statistical model is a seasonal ARIMA conditional mean model (56) with a CCC multivariate GARCH innovation model given by (58) and (59). The above seemingly complex model is extremely efficient to train, as the five parameters ϕ_i, γ_i, α_{i0}, α_{i1} and β_i are learned for each video i separately following the procedures mentioned above, and ρ_{ij} is calculated straightforwardly from recent history.

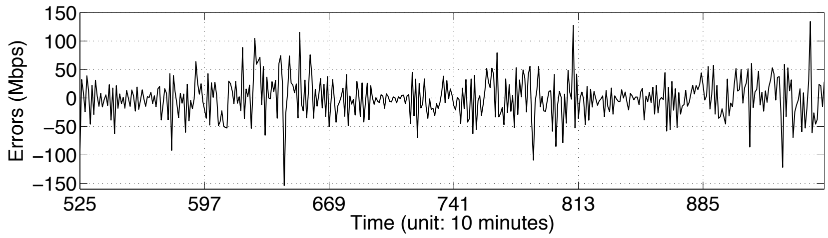
5.1 Model Validation via Real Traces

We verify the effectiveness of the proposed workload prediction models based on the workload traces of UUSee video-on-demand system over a 21-day period during 2008 Summer Olympics [Liu et al. 2010]. As a commercial VoD company, UUSee streams on-demand videos to millions of Internet users across over 40 countries through a downloadable client software. The dataset collected contains performance snapshots taken at a 10-minute frequency of 1693 video channels, including sports events, movies, TV episodes and other genres. The statistics we use in this paper are the time-averaged total bandwidth demand in each video channel in each 10-minute period. There are 144 time periods in a day.

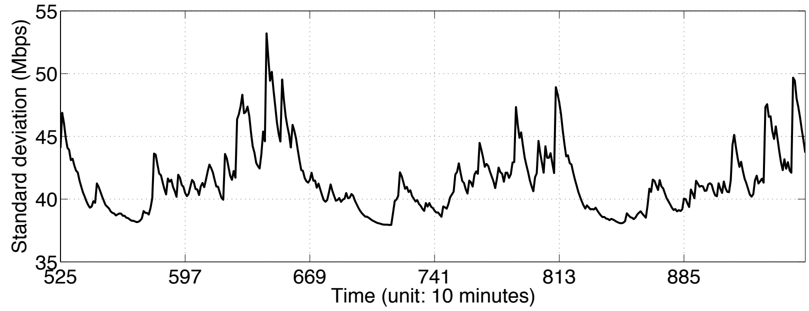
As an example, we make 10-minute-ahead (one-step) prediction of the bandwidth demand of a popular video channel $i = 121$ released at time period $t_0 = 264$ (2008-08-10 10:47:39). The channel has a maximum online population of 2664. The bandwidth consumption series of the first 1.25 days is used as the training data starting from time period 81. The initial 80 time periods are excluded which may not conform to later evolution patterns. The prediction is tested on the data of 3 days following the training period. We fit the low-order models (56) and (58) to the training data and obtain model parameters through a maximum likelihood estimator [Box et al. 2008]. As shown in Fig. 4, such a low-order model merely trained based on the data of 1.25 days can yield conditional mean predictions that are close to the actual demand. The resulted prediction errors plotted in Fig. 4(b), with a mean of zero, have a varying conditional standard deviation predicted by the GARCH model in Fig. 4(c).



(a) One-step conditional mean demand prediction



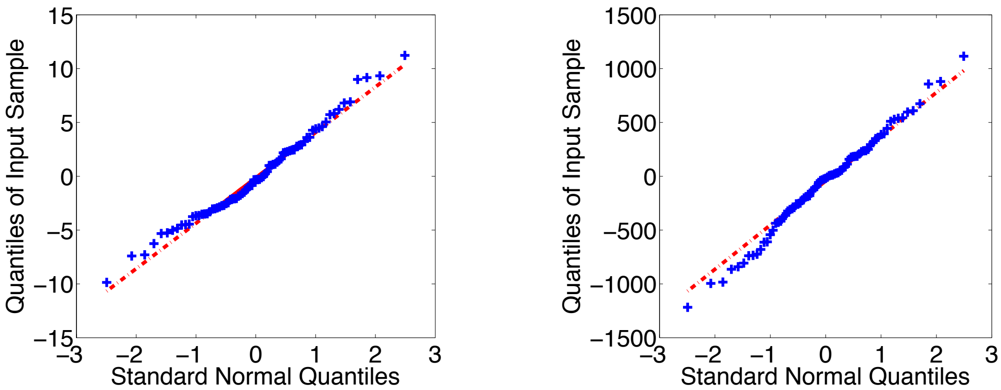
(b) Prediction errors



(c) One-step prediction for conditional standard deviation

Fig. 4: 10-minute-ahead (one-step) prediction for the bandwidth demand of a popular video channel $i = 121$.

Then, we verify that D_{it} approximately follows a Gaussian distribution in each 10-minute period. Recall that for each channel i, given conditional mean prediction $\hat{\mu}_{it}$ at time t, the innovation is $Z_{it} := D_{it} - \hat{\mu}_{it}$. Fig. 5(a) shows the QQ plot of Z_{it} for a typical channel $i = 121$, which indicates $\{Z_{it}\}$ sampled at 10-minute intervals is a Gaussian process. Thus, it is reasonable to assume D_{it} follows a Gaussian distribution within the 10 minutes following t, with mean $\hat{\mu}_{it}$. Fig. 5(b) shows the QQ plot of $\sum_i Z_{it}$, which indicates that the aggregated demand $\sum_i D_{it}$ tends to Gaussian even if D_{jt} is not for some channel j. Since the load L_s of each server is aggregated from many videos, it is reasonable to assume L_s is Gaussian.



(a) Z_{it} of a typical channel

(b) $\sum_i Z_{it}$ of all channels combined

Fig. 5: QQ plot of innovations for $t = 1562$—1640 vs. normal distribution.

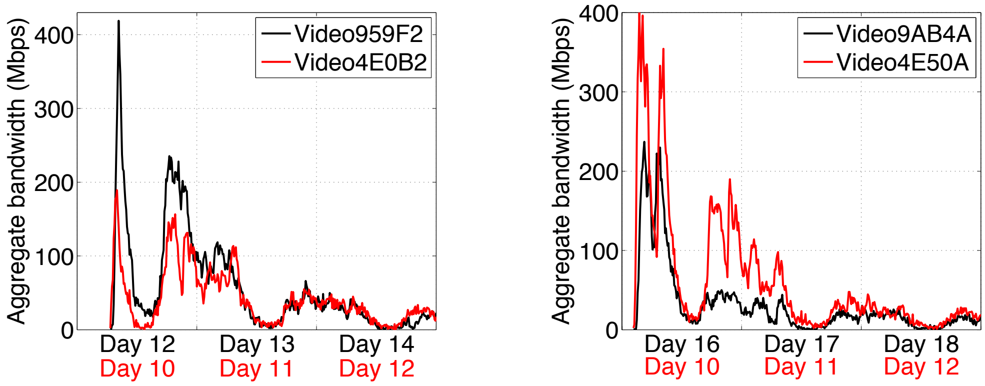


Fig. 6: Videos released on different days but at the same time of day exhibit similar initial demand patterns.

5.2 A Channel Interleaving Scheme

Although we have presented a complete framework for efficient forecasts of expected future demand μ_t and demand covariance matrix $\mathbf{\Sigma}_t$, the parameter learning for the seasonal ARIMA model (56) requires a training data of more than 1 day (specifically 1.25 days in our predictor) to incorporate daily periodicity into the model. As new videos do not have enough historical observations for model training, their demands can hardly be forecasted from history. In this section, we propose methods to predict demands for newly released videos that lack historical observations and unpopular small video channels. We tackle this issue by intelligently interleaving traffic of new videos to form "virtualized video channels" for demand prediction. We also use a similar technique to combine small channels to improve prediction accuracy.

Let us consider new videos that have been in the system for less than 1.25 days. Although these videos do not have sufficient historical observations for model training, we observe that their initial demand patterns are quite similar to videos that were released earlier around *the same time of day*. For example, the left half of Fig. 6 shows the initial demands of 2 video channels 959F2 and 4E0B2 released at 2008-08-19 21:31:56 and 2008-08-17 21:23:20, respectively. As both videos are released around the same time of day, though on different days, they are aligned in Fig. 6 for comparison, with double lines of x-labels showing the first 3 days of each video. (2008-08-08 is deemed as Day 1 and the first time period of each day is 14:50.) We can see the two videos exhibit a similar initial demand evolution pattern, though with different popularity. The major reason for such similarity is that most users watch VoD channels around several peak times in a day: both videos are released between 21:00 and 22:00 and will expect the first peak demand at midnight, followed by a second peak at noon on the next day. Similarly, the right half of Fig. 6 compares the initial demands of video 9AB4A released at 2008-08-23 17:11:54 and video 4E50A released at 2008-08-17 17:02:38. They also exhibit similar initial demand patterns, with the first peak around 18:00, which is the start of off-work hours, before the second peak around midnight. Different videos, however, may attract different sizes of population depending on their popularity.

From the above analysis, we can predict the demand for a new video based on other videos released on an earlier date but at the same time of day. To implement this idea, we define **virtual new channel** k as a combination of all video channels with an age less than 1.25 days and released in hour $k \in \{1, \ldots, 24\}$ on any date. Upon release, a new video joins virtual new channel k based on its release hour k, and quits this virtual new channel when it has been in the system for 1.25 days and accumulated enough observations for separate model training. As a result, each virtual new channel k contains a dynamic set of video channels released in hour k yet possibly on different days. For example, Fig. 7 shows the aggregate bandwidth demand of virtual new channel 11 from time 433 to 1800, and Fig. 9 shows the number of videos contained in virtual new channel 11 from time 1 to 1800. We can see that although virtual new channel 11 represents a dynamic group of videos, its aggregate bandwidth demand exhibits repetition of a similar pattern because the videos in this virtual channel are all released in hour 11, possibly on different dates.

Similarly, we aggregate small video channels and set up 24 **virtual small channels**. When a video reaches the age of 1.25 days, it quits its virtual new channel. If its demand never exceeded a threshold (e.g., 40 Mbps) in the first 1.25 days, it will join one of the virtual small channels in a round robin fashion. Otherwise, it becomes a **mature channel**.

Each mature or virtual channel is deemed as an entity to which predictions and optimizations are applied. For example, we make 10-minutes-ahead prediction of bandwidth demand for virtual new channel 11, and plot the conditional mean prediction in Fig. 7 and the conditional standard deviation prediction in Fig. 8 for a test period of 1.5 days. Satisfactory prediction performance is observed. Although conditional mean prediction is subject to errors, the GARCH model can predict the conditional error standard deviation, as shown in Fig. 8, which contributes to the risk factor (11) in the bandwidth reservation minimization. Furthermore, the combination of several real video channels into a virtual channel suppresses random shocks, making prediction more accurate.

6. PERFORMANCE EVALUATION

We conduct a series of simulations to evaluate the performance of our auto-scaling reservation schemes for video storage systems. The simulations are driven by the replay of the workload traces of UUSee video-on-demand system over a 21-day period during 2008 Summer Olympics [Liu et al. 2010]. We ask the question—what the performance would have been if UUSee had all its workload in this period served by cloud services through our auto-scaled bandwidth reservation system?

We conduct performance evaluation for 4 typical time spans which are near the beginning, middle and end of the 21-day duration. We implement statistical learning and demand prediction techniques presented in Sec. 5 to forecast the expected demands $\boldsymbol{\mu}_t$ and demand covariance matrix $\boldsymbol{\Sigma}_t$ every 10 minutes. The model parameters are retrained daily, with training data being the bandwidth demand

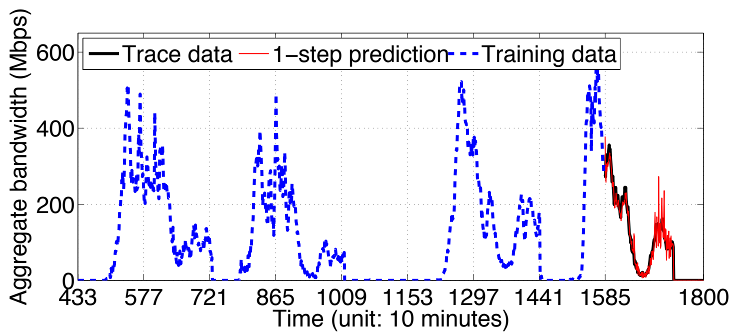


Fig. 7: The conditional mean prediction for S_t in virtual new channel 11, with a test period of 1.5 days from time 1585 to 1800. Only a part of the entire training data is plotted.

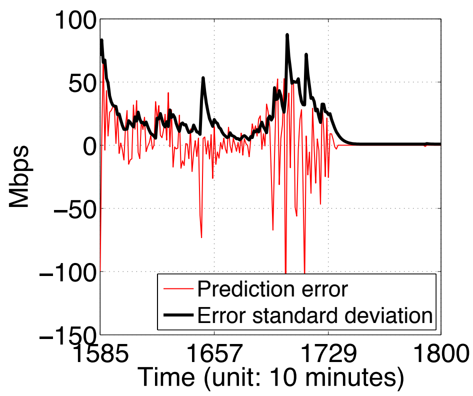


Fig. 8: The prediction error and predicted error standard deviation for S_t in virtual new channel 11.

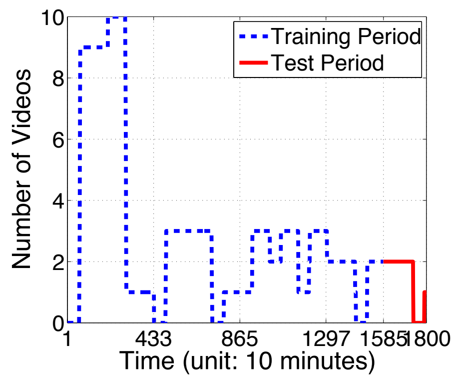


Fig. 9: The number of videos in virtual new channel 11 during the entire training and test periods.

series $\{D_{i\tau}\}$ in the recent 1.25 days of each channel i. Once trained, the models will be used for the next 24 hours. Although video users may join or quit a channel unexpectedly, our prediction is still effective, since it deals with the *aggregate demand* in the channel which features diurnal patterns. We assume that there is a pool of servers from which UUSee can reserve bandwidth. To spread the load across servers, we set $C_s = 300$ Mbps for each s. The QoS parameter $\theta := F^{-1}(1-\epsilon)$ is set to $\theta = 2.05$ to confine the under-provision probability to $\epsilon = 2\%$.

6.1 Algorithms for Comparison

We compare our optimal load direction (14) under full replication, and Algorithm 1, Algorithm 2, Algorithm 3 and Algorithm 4 with sparse content placement, against the following baseline algorithms:

Reactive without Prediction. Initially, replicate each video to K randomly chosen servers, which limits the initial content replication degree to K. Each client requesting channel i is randomly directed to a server that has video i and idle bandwidth capacity. A request is dropped if there is no such server. In this case, the algorithm reacts by replicating video i to an additional server chosen randomly that has idle capacity. Replicating content is not instant: we assume that the replication involves a delay of one period of time.

Table I. : The performance of different schemes averaged over each test period, in terms of QoS, resource utilization, and replication.

Periods	**Time periods 702—780** (91 mature and virtual channels) Peak demand 6.56 Gbps, mean demand 5.19 Gbps						**Time periods 1422—1480** (161 mature and virtual channels) Peak demand 6.81 Gbps, mean demand 4.91 Gbps					
	Short	Drop	Util	Rep	Booked	Over-prov	Short	Drop	Util	Rep	Booked	Over-prov
Optimal	0 Chs	0.66%	92.9%	91.0	6.57 Gbps	108.5%	0 Chs	0.25%	91.1%	161.0	6.38 Gbps	110.3%
Per-Server Opt	1.0 Chs	0.37%	90.0%	8.5	6.79 Gbps	112.2%	1.2 Chs	0.13%	88.6%	6.9	6.56 Gbps	113.4%
Per-Server Lim	0.3 Chs	0.06%	85.7%	2.6	7.13 Gbps	117.8%	0.2 Chs	0.03%	84.6%	2.4	6.86 Gbps	118.8%
Random	5.9 Chs	0.02%	83.3%	3.8	7.33 Gbps	121.2%	7.6 Chs	0.00%	82.2%	3.0	7.08 Gbps	122.4%
Reactive	7.9 Chs	0.47%	77.2%	4.3	7.91 Gbps	132.4%	7.2 Chs	0.34%	70.4%	3.6	8.20 Gbps	146.0%
Itr L_1-Constr	0 Chs	0.18%	88.2%	4.8	6.92 Gbps	114.3%	-	-	-	-	-	-
Itr L_1-Penal	0.1 Chs	0.06%	85.1%	2.3	7.18 Gbps	118.7%	0.1 Chs	0%	84.7%	2.3	6.88 Gbps	118.8%
Periods	**Time periods 1562—1640** (176 mature and virtual channels) Peak demand 7.55 Gbps, mean demand 5.62 Gbps						**Time periods 2402—2500** (199 mature and virtual channels) Peak demand 9.19 Gbps, mean demand 7.62 Gbps					
	Short	Drop	Util	Rep	Booked	Over-prov	Short	Drop	Util	Rep	Booked	Over-prov
Optimal	0 Chs	0.31%	91.1%	176.0	7.27 Gbps	110.4%	0 Chs	0.11%	85.4%	199.0	10.54 Gbps	118.1%
Per-Server Opt	0.7 Chs	0.16%	88.3%	7.3	7.51 Gbps	114.0%	1.0 Chs	0.09%	82.7%	6.3	10.87 Gbps	121.8%
Per-Server Lim	1.4 Chs	0.00%	83.9%	2.4	7.89 Gbps	119.9%	20.7 Chs	0.17%	82.3%	2.5	10.95 Gbps	122.6%
Random	6.2 Chs	0.00%	80.4%	3.3	8.28 Gbps	125.4%	33.4 Chs	0.02%	77.9%	4.5	11.54 Gbps	129.3%
Reactive	5.9 Chs	0.27%	72.7%	3.5	9.08 Gbps	140.4%	15.8 Chs	0.43%	74.6%	3.6	12.01 Gbps	140.3%
Itr L_1-Penal	1.1 Chs	0.03%	84.5%	2.0	7.85 Gbps	119.1%	1.0 Chs	0.01%	81.1%	2.1	11.92 Gbps	124.5%

Short: Average # channels with dropped requests; **Drop**: average request drop rate; **Util**: average utilization of allocated resources; **Rep**: average replication degree; **Booked**: average booked bandwidth; **Over-prov**: average over-provisioning ratio.

Note: Iterative L_1-Constrained is only evaluated for time periods 702-780, since it cannot efficiently complete within 10 minutes for more than 91 channels, which is the case for other time spans.

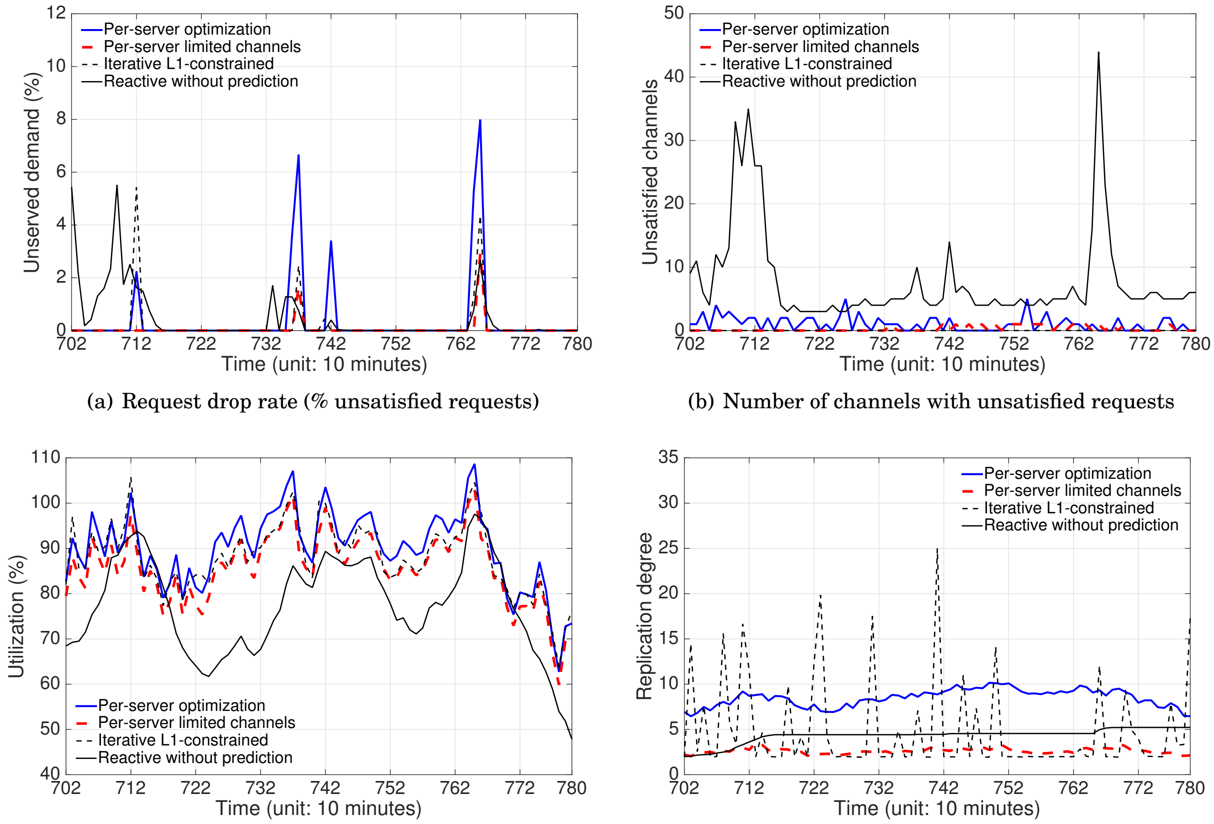
Random with Prediction. Initially, let $s = 1$ and $\mathbf{b} = \mathbf{1}$. Second, randomly generate $\mathbf{w}_s$ in $(\mathbf{0}, \mathbf{b})$ and rescale it so that the QoS constraint (11) is achieved with equality for s. Update $\mathbf{b}$ to $\mathbf{b} - \mathbf{w}_s$ and update s to $s+1$. Go to the second step unless $\mathbf{b} = 0$ or $s = S+1$, in which case the program terminates.

The reactive scheme represents provisioning for peak demand in Fig. 1 in some way, with limited replication. It does not leverage prediction or bandwidth reservation. We assume in Reactive, the total cloud capacity allocated is always the minimum capacity needed to meet the peak demand in the system. The random scheme leverages prediction and makes bandwidth reservation, but randomly directs workloads instead of using anti-correlation and optimization techniques to minimize bandwidth reservation.

We implement all of the six schemes discussed above, and summarize their performance comparison in Table I for each of the four time spans. Iterative L_1-Constrained is only evaluated for time periods 702-780, as it cannot converge within 10 minutes for more than 91 channels. Note that the channels in the table include mature channels, virtual new channels and virtual small channels. The number of videos in each virtual channel can vary over time. As new videos are introduced, more channels are present in later test periods. We evaluate the algorithm performance with regard to QoS, bandwidth resource occupied, and replication cost.

6.2 The Benefit of Predictive Provisioning over Reactive Provisioning

Table I shows that Reactive generally has a more salient QoS problem than all five predictive schemes in terms of both the number of unsatisfied channels and request drop rate (percentage of unsatisfied requests), demonstrating the benefit of demand prediction. Fig. 10 presents a more detailed comparison for a typical peak period from time 702 to 780. Without surprise, Reactive has many unfulfilled requests at the beginning. Since the videos are randomly replicated to $K = 2$ servers (shown in Fig. 10(d) at $t = 702$) and requests are randomly directed, it is likely that a channel does not acquire enough ca-



(a) Request drop rate (% unsatisfied requests)

(b) Number of channels with unsatisfied requests

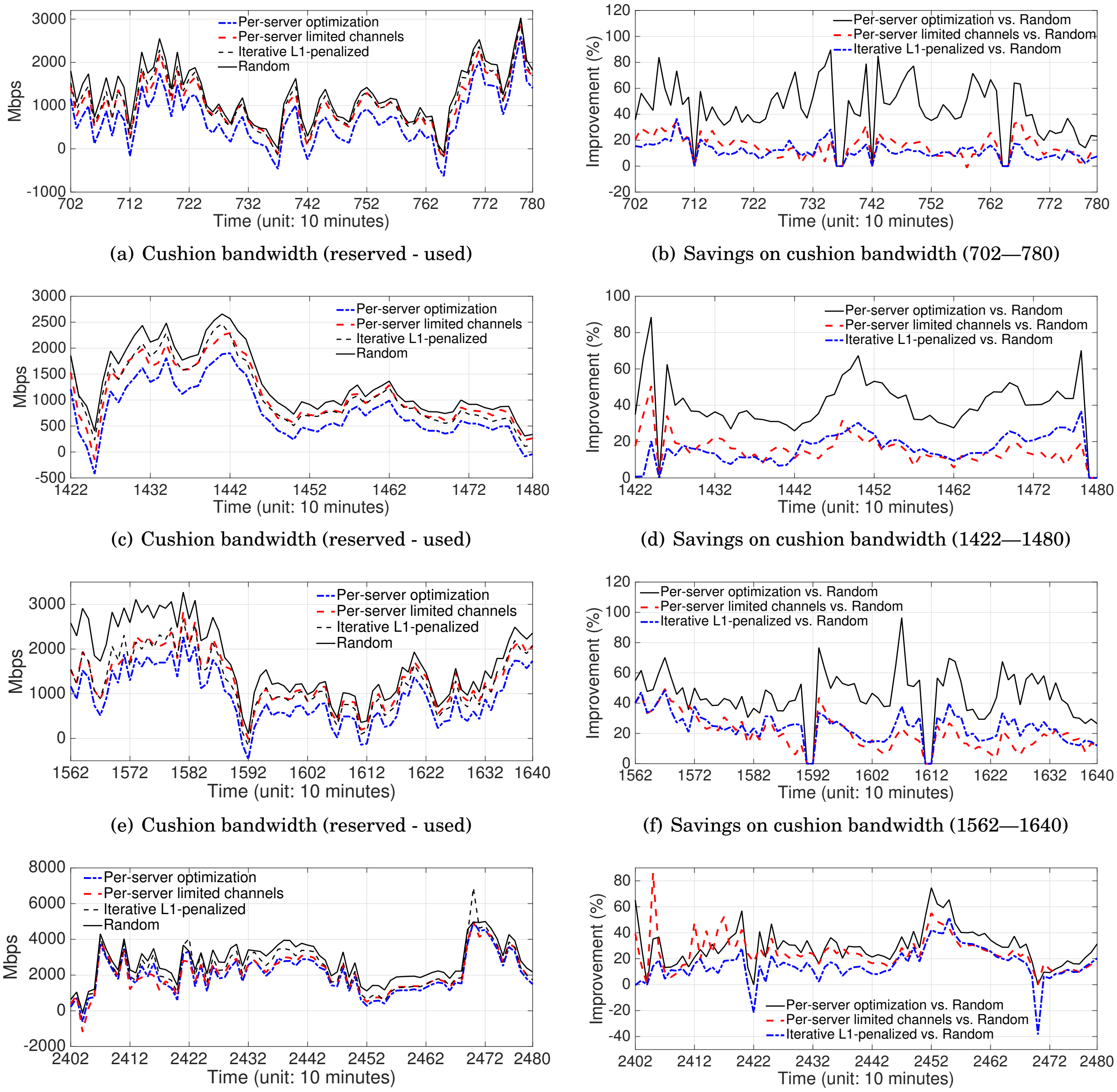
(c) Utilization of reserved resource

(d) Replication degree

Fig. 10: Predictive vs. reactive bandwidth provisioning for a typical peak period 702–780. There are 35 servers available, each with capacity 300 Mbps, and 91 channels, including 52 popular channels, 24 small channels, 15 non-zero new channels. For Reactive, $K = 2$. For Iterative L_1-Constrained, the number of videos per server is $k_s = 5$ for all s. For all other schemes, $k_s = 10$ for all s.

pacity to meet its demand. As Reactive detects the QoS problem, videos are replicated to more servers to acquire more capacity, with a gradually increased replication degree over time, as in Fig. 10(d). We can see that after 140 minutes, when the replication degree exceeds 4, the QoS of Reactive becomes relatively stable in Fig. 10(a). However, around time 763, Reactive suffers from salient QoS problems again, due to a sudden ramp-up of demand. In contrast, the predictive schemes foresee and get prepared for demand changes, resulting in much better QoS, even in the event of drastic demand increase.

The predictive schemes also achieve higher resource utilization. Utilization of a predictive scheme is the ratio between the actual bandwidth usage and the total booked bandwidth in all servers. For Reactive, the utilization is the actual bandwidth demand divided by the peak demand. Although Fig. 10(c) shows that Reactive achieves a high utilization for the peak demand around time 763, its average utilization is merely 77.19% in the test period from 702 to 780. Predictive auto-scaling enhances utilization to 85.7% with Per-Server Limited Channels, to 90.0% with Per-Server Optimal, to 88.2% with Iterative L_1-Constrained and to 92.9% with the theoretical optimal solution under full replication.



(a) Cushion bandwidth (reserved - used)

(b) Savings on cushion bandwidth (702—780)

(c) Cushion bandwidth (reserved - used)

(d) Savings on cushion bandwidth (1422—1480)

(e) Cushion bandwidth (reserved - used)

(f) Savings on cushion bandwidth (1562—1640)

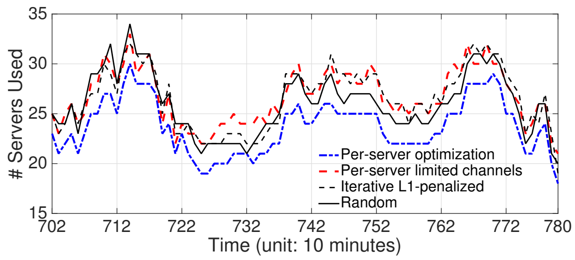
(g) Cushion bandwidth (reserved - used)

(h) Savings on cushion bandwidth (2402—2480)

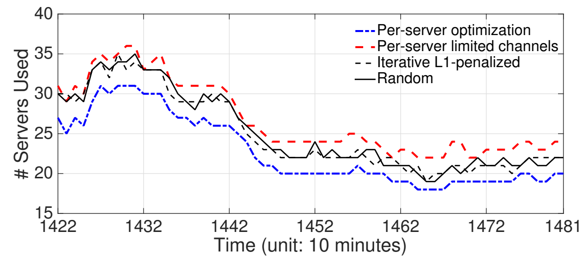
Fig. 11: Workload portfolio selection vs. random load direction for different time periods. For all the schemes, the number of videos per server is $k_s = 10$ for all s.

6.3 Resource Autoscaling: a Comparison among Predictive Schemes

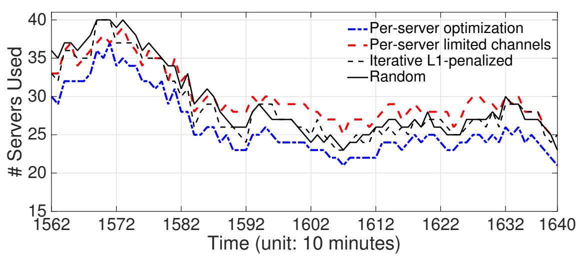
We now compare the six predictive schemes. Among them, as shown in Table I, Optimal books the minimum necessary bandwidth and achieves the highest bandwidth utilization, yet with the highest



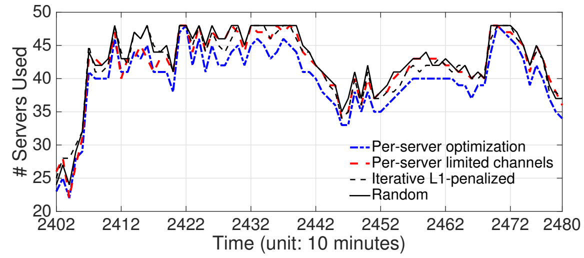
(a) Server autoscaling for time periods 702–780



(b) Server autoscaling for time periods 1422–1480



(c) Server autoscaling for time periods 1562–1640



(d) Server autoscaling for time periods 2402–2500

Fig. 12: Server autoscaling: the number of servers used by each predictive provisioning scheme in each time period for 4 different time spans.

replication overhead. In fact, with full replication, each video is replicated to every server, and thus the optimal solution can best exploit the anti-correlations among all the channels to minimize reserved bandwidth. However, the VoD provider needs to pay a high storage cost to the cloud service provider.

Among all the five predictive schemes that replicate content sparsely, Random achieves the lowest utilization, since it is completely blind to the correlation information in workload selection and direction. Per-Server Optimal can reduce the replication degree while maintaining other performance metrics. By further imposing a channel number constraint on each server, Per-Server Limited Channels strikes a balance between replication overhead and bandwidth utilization. It aggressively reduces the replication degree to a very small value of 2.4-2.6 copies per video. Iterative $L1$-penalized turns out to be a numerically stable method which yields the smallest replication degree among all the predictive schemes, with an extremely low drop rate and an over-provisioning ratio that is only slightly higher than Optimal and comparable to Per-Server Limited Channels.

Nonetheless, Iterative L_1-Constrained, as shown in Fig. 10(c) and Fig. 10(d), achieves a slightly higher utilization of booked bandwidth than Per-Server Limited Channels at the cost of a higher replication degree. The request drop rates and numbers of unsatisfied channels in both schemes are similar to each other, as shown in Fig. 10(a) and Fig. 10(b). Note that for Iterative L_1-Constrained, we have set the number of videos per server to be $k_s = 5$, which is one half of that in other schemes. The reason is that in Iterative L_1-Constrained, the modified constraint $\sum_i \hat{\phi}^t_{si}(w_{si}) \le k_s$ does not always converge to the video number (L_0-norm) constraint per server $\|\mathbf{w}_s\|_0 \le k_s$. In Fig. 10(d), the spikes in the replication degree corresponds to the time periods where the iterative program aborts in an iteration when there is no feasible solution to constraints (39)-(42). In such cases, the modified constraint (42), i.e., $\sum_i \hat{\phi}^t_{si}(w_{si}) \le k_s$ never converges to $\|\mathbf{w}_s\|_0 \le k_s$. Thus, there exist much higher replication degrees in such time periods, although k_s is set to a low value. In fact, it is challenging to tune the parameter δ so that Iterative L_1-Constrained can always converge for all time periods with different input demands. Furthermore, for 91 channels, Iterative L_1-Constrained takes on average 600 seconds[2] to finish the iterative optimization procedure for each time period, which exceeds 10 minutes, the length of each time period. Therefore, Iterative L_1-Constrained is not efficient and stable enough for the purpose of content placement.

In contrast, both Per-Server Limited Channels and Iterative L_1-Penalized are much more efficient and numerically stable: it takes up to only 2 minutes for prediction plus either of the two schemes to finish for each time period, well before the deadline of 10 minutes. Therefore, considering replication degree, QoS, utilization and computational efficiency, Per-Server Limited Channels and Iterative L_1-Penalized are the best, although it will be demonstrated subsequently that Iterative $L1$-penalized has the additional benefit of mitigating content migration overhead across time periods.

We further show a detailed comparison between Per-Server Optimal, PerServer limited channels, Iterative L_1-Penalized, and Random for all 4 time spans in Fig. 11. The efficiency of predictive bandwidth booking can be evaluated by the *cushion bandwidth* needed, which is the gap between the booked bandwidth and actual required bandwidth. Fig. 11(c), Fig. 11(e), and Fig. 11(g) plot the cushion bandwidth. For example, during time periods 1562—1640, while being on the same QoS level, random load direction results into a cushion bandwidth up to 3 Gbps compared to a mean demand of 5.62 Gbps, representing significant over-provisioning. Using Per-Server Optimal, the cushion bandwidth can be saved by 50% on average, as shown in Fig. 11(f). Per-Server Limited Channels and Iterative L_1-Penalized, even with a replication degree of about 2 copies per video, can save cushion bandwidth by around 30% as compared to Random, which has a higher replication degree of 3.3 copies per video.

QoS problems occur if bandwidth is under-provisioned, leading to a cushion bandwidth below 0. For example, from Fig. 11(e), we observe that QoS problems occur occasionally for Per-Server Optimal, but seldom for Per-Server Limited Channels and Iterative L_1-Penalized from time 1562 to 1640, because the latter schemes conservatively book more cushion bandwidth.

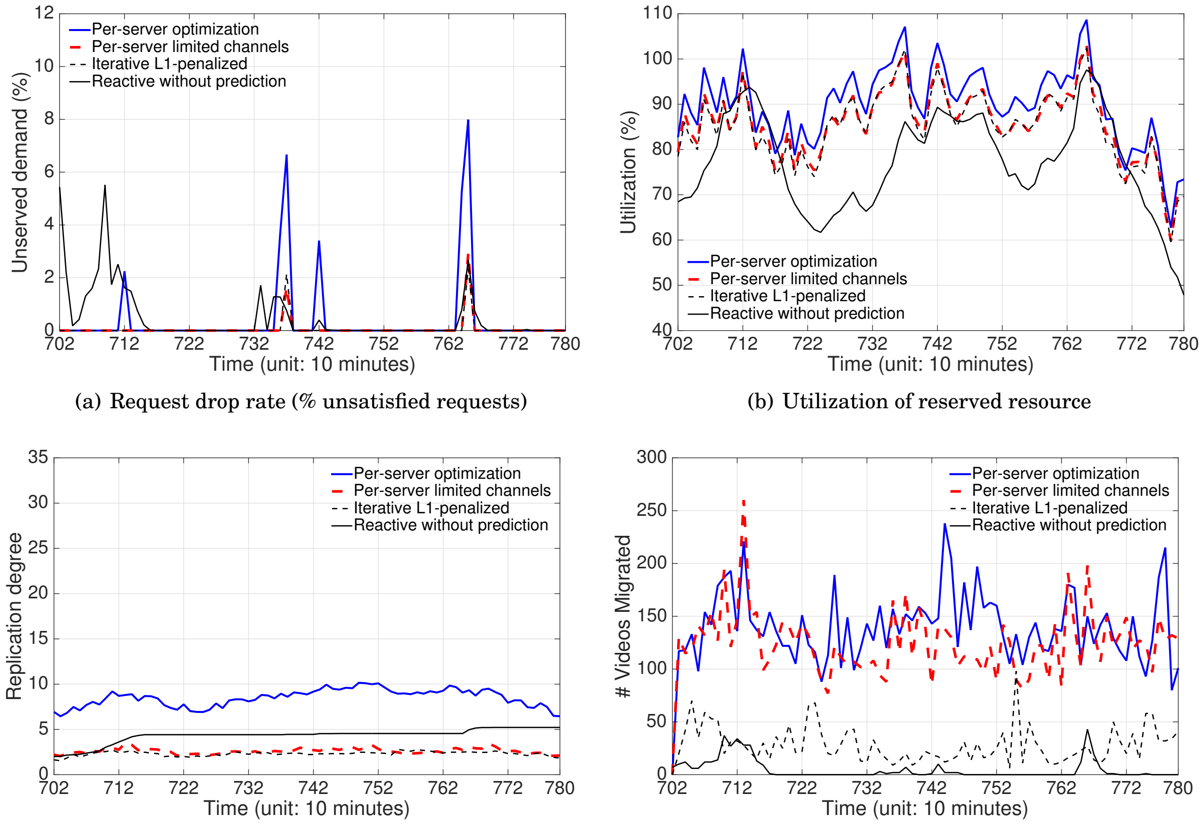
An important advantage of our schemes is that they can autoscale the number servers (instances in terms of cloud computing) used. The actual numbers of servers used by different predictive schemes in different time periods are shown in Fig. 12. Since our algorithms adopt a per-server heuristic, they can push most loads only onto necessary servers instead of letting the load spread across the available server pool. This enables the idle servers to be used for other purposes.

6.4 Replication and Migration Overhead

We now evaluate the replication and migration overhead in the simulated video storage system. We compare the replication degree, migration overhead, QoS and utilization achieved by Per-Server Optimal, Per-Server Limited Channels, Iterative L_1-Penalized and Reactive in all 4 different time spans, and show the results in Figures 13, 14, 15, 16, respectively.

From these figures, we can see that as compared to all other schemes, Iterative L_1-Penalized can effectively reduce the number of video copies transferred in each time period by using a regularizer to limit the difference from the previous placement decision, avoiding the global shuffling. In the meantime, Iterative L_1-Penalized achieves a slightly lower replication degree as Per-Server Limited Channels and a similar level of high resource utilization as Per-Server Limited Channels.

[2]Running times are measured on a 2.6GHz Intel Core i7 processor.



(a) Request drop rate (% unsatisfied requests)

(b) Utilization of reserved resource

(c) Replication degree

(d) Number of video copies migrated

Fig. 13: Performance of different schemes for a typical peak period 702–780. There are 35 servers available, each with capacity 300 Mbps, and 91 mature and virtual channels. For Reactive, $K = 2$. For Per-Server Limited Channels, $k_s = 10$ for all s.

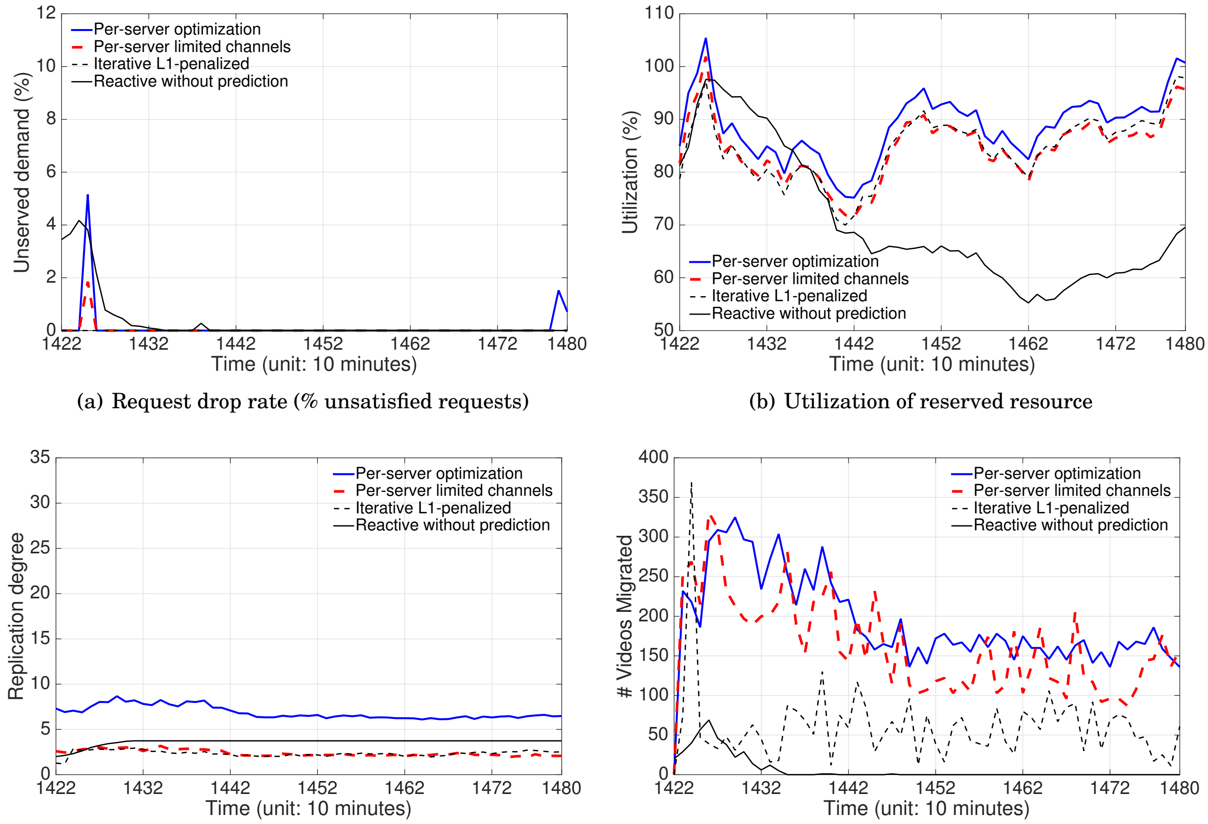
Furthermore, the execution of Iterative L_1-Penalized is quite light-weight in our simulation. In the subroutine, Algorithm 5, we set $maxiteration = 5$, and set $\lambda(1) = 0$ and

$$\lambda(t) = \frac{1}{5} \cdot \frac{\boldsymbol{\mu}^{\mathsf{T}} \mathbf{w}_s^{t-1}}{\sum_{i:w_{si}^{\text{pre}}=0} \hat{\phi}_{si}^{t}(w_{si}^{t-1})}, \quad t = 2, 3, 4, 5.$$

With the above setting, it takes less than 1 minute to execute Iterative L_1-Penalized and the solution is already sparse enough.

7. RELATED WORK

Researches on exploiting virtualization techniques for delivering cloud-based IPTV services have been conducted by major VoD providers like AT&T [Aggarwal et al. 2011]. The importance of VoD bandwidth demand prediction to capacity planning has also been recognized. It is shown that demand estimates can help with optimal content placement in AT&T's IPTV network [Applegate et al. 2010].



(a) Request drop rate (% unsatisfied requests)

(b) Utilization of reserved resource

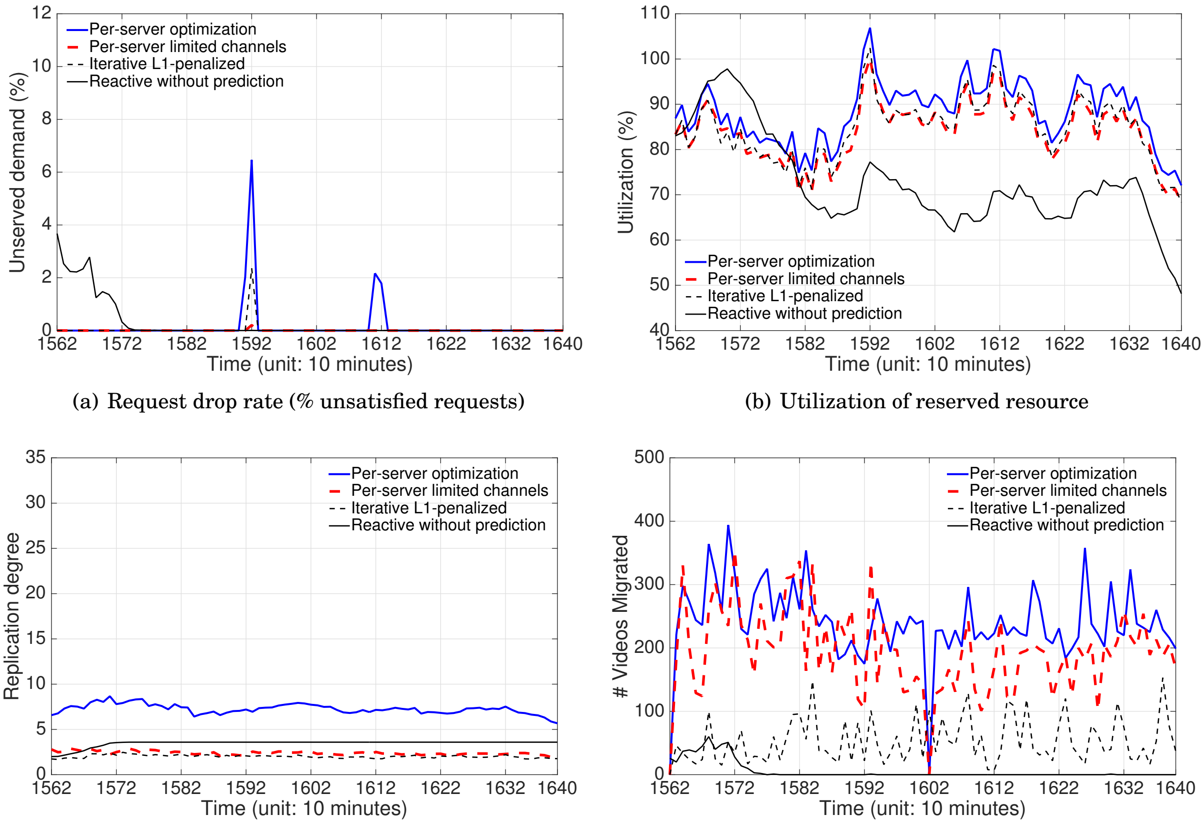
(c) Replication degree

(d) Number of video copies migrated

Fig. 14: Performance of different schemes for a typical peak period 1422–1480. There are 36 servers available, each with capacity 300 Mbps, and 161 mature and virtual channels. For Reactive, $K = 2$. For Per-Server Limited Channels, $k_s = 10$ for all s.

The traffic characteristics of the two popular video streaming services, Netflix and YouTube, are studied in [Rao et al. 2011], where it is observed that the bandwidth of links carrying video streaming traffic should be provisioned to $\mathbf{E}[R(t)] + \alpha\sqrt{\mathbf{Var}[R(t)]}$, where $R(t)$ is the aggregate data rate of the video streaming traffic at time t. $\alpha \geq 1$ is a constraint on the tolerable bandwidth violations. Further more, it is pointed out in [Rao et al. 2011] that the mean and variance of the aggregate data rate of video streaming traffic are independent of the underlying streaming strategies used, which may range from non-ack clocked ON-OFF cycles to bulk TCP transfer, depending on the type of the application (Web browser or native mobile application) and the type of container (Silverlight, Flash, or HTML5) used. Hence, the required bandwidth is also independent of these diverse factors. This implies that video services can safely select a streaming strategy that can be optimized for other goals such as server load without overwhelming the network.

Due to the predictability of aggregate video traffic, several time-series and statistical learning methods have been applied to video traffic prediction, including non-stationary time series models [Niu et al. 2011b], [Niu et al. 2011a], and video access pattern extraction via principal component analysis [Gürsun et al. 2011].



(a) Request drop rate (% unsatisfied requests)

(b) Utilization of reserved resource

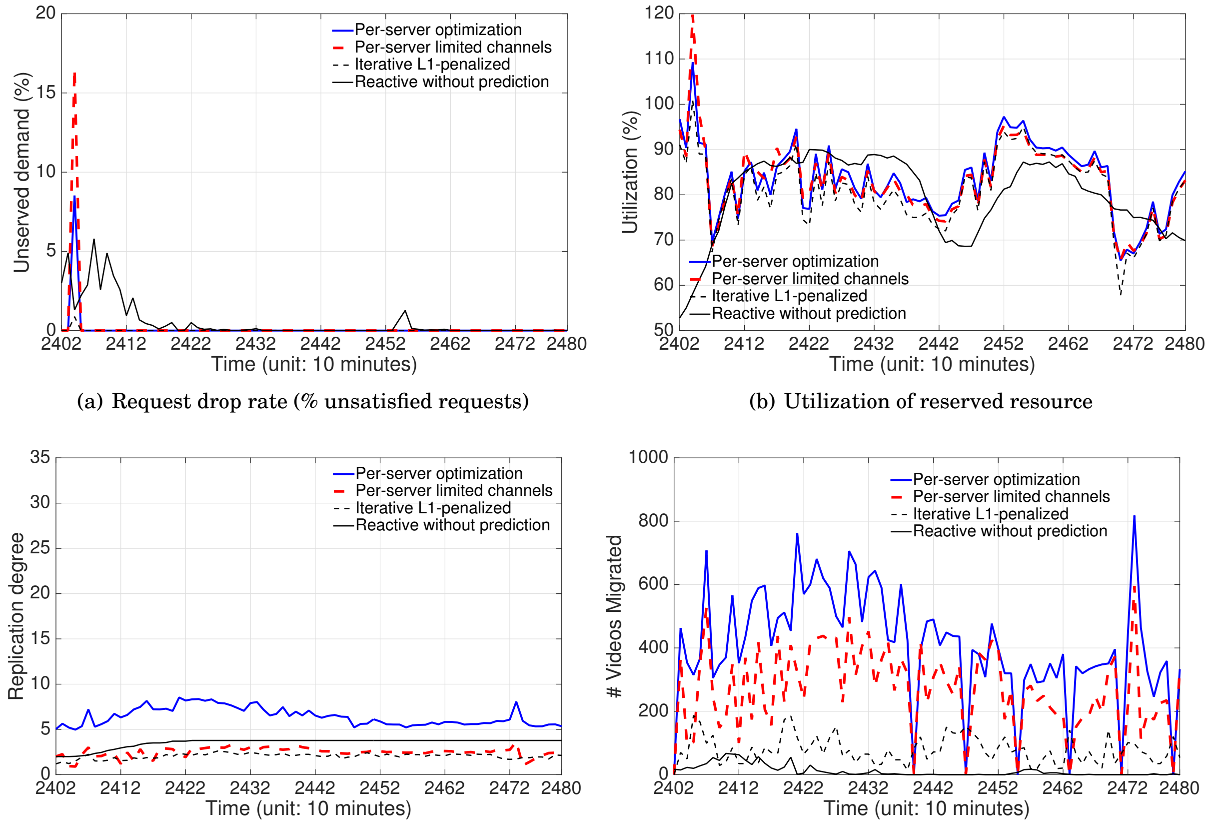
(c) Replication degree

(d) Number of video copies migrated

Fig. 15: Performance of different schemes for a typical peak period 1562–1640. There are 40 servers available, each with capacity 300 Mbps, and 176 mature and virtual channels. For Reactive, $K = 2$. For Per-Server Limited Channels, $k_s = 10$ for all s.

Predictive and dynamic resource provisioning has been proposed mostly for virtual machines (VM) and web applications with respect to CPU utilization [Bobroff et al. 2007], [Gong et al. 2010], [Tang et al. 2007], [Gmach et al. 2007] and power consumption [Kusic et al. 2009], [Lin et al. 2011]. VM consolidation with dynamic bandwidth demand has also been considered in [Wang et al. 2011]. Our work exploits the unique characteristics of VoD bandwidth demands and distinguishes from the above work in three aspects. *First*, our bandwidth workload consolidation is as simple as solving convex optimization for a load direction matrix. We leverage the fact that unlike VM, demand of a VoD channel can be *fractionally* split into video requests. *Second*, our system forecasts not only the expected demand but also the demand volatility, and thus can control the risk factors more accurately. In contrast, most previous works [Gong et al. 2010], [Gmach et al. 2007] assume a constant demand variance. *Third*, we exploit the statistical correlation between bandwidth demands of different video channels to save resource reservation while previous works such as [Wang et al. 2011] consider independent workloads.

The idea of statistical multiplexing and resource overbooking has been empirically evaluated for a shared hosting platform in [Urgaonkar et al. 2002]. Our novelty is that we formulate the quality-assured resource minimization problem using Value at Risk (VaR), a useful risk measure in financial



(a) Request drop rate (% unsatisfied requests)

(b) Utilization of reserved resource

(c) Replication degree

(d) Number of video copies migrated

Fig. 16: Performance of different schemes for a typical peak period 2402–2480. There are 48 servers available, each with capacity 300 Mbps, and 199 mature and virtual channels. For Reactive, $K = 2$. For Per-Server Limited Channels, $k_s = 10$ for all s.

asset management [McNeil et al. 2005], with the aid of accurate demand correlation forecasts. We believe our theoretically grounded approach bears stronger robustness against intractable demand volatility in practice.

There are extensive studies around the content placement problem in replication-based cloud storage systems. In [Rochman et al. 2013], Rochman et al. propose the strategies of placing the resources to distributed datacenters to serve more requests locally. In [Xu and Li 2013], Xu et al. propose a request mapping and response routing scheme to maximize the total utility of serving requests minus the cost. Bonvin et al. [Bonvin et al. 2010] propose a distributed scheme to dynamically allocate the resources of a data storage cloud based on net benefit maximization, considering the utility offered by the partition and its storage and maintenance cost. In [Agarwal et al. 2010], automatic data placement across geo-distributed datacenters is presented, which iteratively moves a data item closer to both clients and other data items that it communicates with. B. Yang et al. [Yu and Pan 2015] study the content placement problem for systems when multiple items are needed in each request and the item size is small. They try to maximize the correlation of the items collocated on the same server to reduce the I/O and CPU overhead to satisfy each request. In this paper, we consider video storage and access systems, where the most important performance metrics are bandwidth and storage. And we focus on geographically collocated server clusters in a same datacenter.

Our prior work [Niu et al. 2012] has studied the optimal load direction and cloud bandwidth reservation under full content replication, this paper provides a deeper study on this problem in a wider scope. Specifically, in this paper, we further discuss load direction under a given sparse replication scheme and study the joint optimization of load direction and sparse content placement in Sec. 4. We introduce a new algorithm to solve this joint load direction and placement problem involving L_0 norms through iteratively reweighed L_1-norm relaxations, and compare its performance with other proposed algorithms. Furthermore, we elaborate our demand prediction schemes in Sec. 5, with detailed performance evaluations, and add detailed discussions on handling new channels and forming virtual channels for improved demand prediction.

To the best of our knowledge, this is the first work that jointly models the load direction and sparse content placement as a sparsity-penalized or sparsity-constrained optimization problem. And we have novelly adapted the iterative reweighted $L1$-norm approximation techniques from the sparse recovery theory to solve our sparse design problem, yielding satisfactory performance and low computational complexity.

8. CONCLUDING REMARKS

In this paper, we propose an unobtrusive, predictive, and elastic cloud resource auto-scaling framework for video storage systems. Operated at a 10-minute frequency, the system automatically predicts the expected future demand as well as demand volatility for each video through ARIMA and GARCH time-series forecasting techniques based on history. Leveraging demand prediction, the system jointly makes load direction to multiple cloud servers and make bandwidth reservations from them to satisfy the projected demands with high probability. The system can save resource reservation cost for VoD providers in terms of both bandwidth and storage.

We exploit the predictable anti-correlation between video requests to enhance resource utilization, and derive the optimal load direction that minimizes bandwidth resource reservation while confining under-provision risks. We formulate the joint load direction and sparse content placement problem as an L_0-norm regularized optimization which turns to be nonconvex. To approximately solve this problem, we propose, among several heuristics, an iteratively reweighted L_1-norm penalized optimization process that can yield sparse placement and reduce content migration.

Based on extensive simulations driven by the demand traces of a large-scale production VoD system, we observe that the proposed Iterative L_1-Penalized optimization has the best practical appeals due to its capability of efficiently computing solutions that can balance the costs of bandwidth and storage with limited migration overhead, while achieving satisfying quality of service.

REFERENCES

Amazon Web Services. http://aws.amazon.com/.

AGARWAL, S., DUNAGAN, J., JAIN, N., SAROIU, S., WOLMAN, A., AND BHOGAN, H. 2010. Volley: Automated data placement for geo-distributed cloud services. In *NSDI*. 17–32.

AGGARWAL, V., CHEN, X., GOPALAKRISHNAN, V., JANA, R., RAMAKRISHNAN, K. K., AND VAISHAMPAYAN, V. A. 2011. Exploiting Virtualization for Delivering Cloud-based IPTV Services. In *Proc. IEEE INFOCOM Workshop on Cloud Computing*.

APPLEGATE, D., ARCHER, A., LEE, V. G. S., AND RAMAKRISHNAN, K. 2010. Optimal Content Placement for a Large-Scale VoD System. In *Proc. ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*.

BALLANI, H., COSTA, P., KARAGIANNIS, T., AND ROWSTRON, A. 2011. Towards Predictable Datacenter Networks. In *Proc. of SIGCOMM'11*. Toronto, ON, Canada.

BARI, M. F., BOUTABA, R., ESTEVES, R., GRANVILLE, L. Z., PODLESNY, M., RABBANI, M. G., ZHANG, Q., AND ZHANI, M. F. 2013. Data center network virtualization: A survey. *Communications Surveys & Tutorials, IEEE 15,* 2, 909–928.

BOBROFF, N., KOCHUT, A., AND BEATY, K. 2007. Dynamic Placement of Virtual Machines for Managing SLA Violations. In *Proc. 10th IFIP/IEEE International Symposium on Integrated Network Management*.

BOLLERSLEV, T. 1986. Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics 31*, 307–327.

BONVIN, N., PAPAIOANNOU, T. G., AND ABERER, K. 2010. A self-organized, fault-tolerant and scalable replication scheme for cloud storage. In *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 205–216.

BOX, G. E. P., JENKINS, G. M., AND REINSEL, G. C. 2008. *Time Series Analysis: Forecasting and Control*. Wiley.

CANDES, E. J., WAKIN, M. B., AND BOYD, S. P. 2008. Enhancing sparsity by reweighted 1 minimization. *Journal of Fourier analysis and applications 14,* 5-6, 877–905.

ENDERS, W. 2010. *Applied Econometric Time Series* Third Ed. Wiley, Hoboken, NJ.

FAZEL, M., HINDI, H., AND BOYD, S. 2003. Log-det Heuristic for Matrix Rank Minimization with Applications to Hankel and Euclidean Distance Matrices. In *Proc. of American Control Conference*.

GMACH, D., ROLIA, J., CHERKASOVA, L., AND KEMPER, A. 2007. Workload Analysis and Demand Prediction of Enterprise Data Center Applications. In *Proc. IEEE Symp. Workload Characterization*.

GONG, Z., GU, X., AND WILKES, J. 2010. PRESS: PRedictive Elastic ReSource Scaling for Cloud Systems. In *Proc. IEEE International Conference on Network and Services Management (CNSM)*.

GUO, C., LU, G., WANG, H. J., YANG, S., KONG, C., SUN, P., WU, W., AND ZHANG, Y. 2010. SecondNet: a Data Center Network Virtualization Architecture with Bandwidth Guarantees. In *Proc. ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*.

GÜRSUN, G., CROVELLA, M., AND MATTA, I. 2011. Describing and Forecasting Video Access Patterns. In *Proc. IEEE INFOCOM Mini-Conference*.

KUSIC, D., KEPHART, J. O., HANSON, J. E., KANDASAMY, N., AND JIANG, G. 2009. Power and Performance Management of Virtualized Computing Environments via Lookahead Control. *Cluster computing 12,* 1, 1–15.

LIN, M., WIERMAN, A., ANDREW, L. L. H., AND THERESKA, E. 2011. Dynamic Right-Sizing for Power-Proportional Data Centers. In *Proc. IEEE INFOCOM*.

LIU, Z., WU, C., LI, B., AND ZHAO, S. 2010. UUSee: Large-Scale Operational On-Demand Streaming with Random Network Coding. In *Proc. IEEE INFOCOM*.

MCNEIL, A., FREY, R., AND EMBRECHTS, P. 2005. *Quantitative Risk Management: Concepts Techniques and Tools*. Princeton University Press.

NETFLIX. 2010. Four Reasons We Choose Amazon's Cloud as Our Computing Platform. *The Netflix "Tech" Blog*.

NIU, D., LI, B., AND ZHAO, S. 2011a. Understanding Demand Volatility in Large VoD Systems. In *Proc. the 21st International workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*.

NIU, D., LIU, Z., LI, B., AND ZHAO, S. 2011b. Demand Forecast and Performance Prediction in Peer-Assisted On-Demand Streaming Systems. In *Proc. IEEE INFOCOM Mini-Conference*.

NIU, D., XU, H., LI, B., AND ZHAO, S. 2012. Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In *Proc. of IEEE INFOCOM 2012*. Orlando, Florida.

RAO, A., LEGOUT, A., LIM, Y.-S., TOWSLEY, D., BARAKAT, C., AND DABBOUS, W. 2011. Network characteristics of video streaming traffic. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*. ACM, 25.

ROCHMAN, Y., LEVY, H., AND BROSH, E. 2013. Resource placement and assignment in distributed network topologies. In *INFOCOM, 2013 Proceedings IEEE*. IEEE, 1914–1922.

TANG, C., STEINDER, M., SPREITZER, M., AND PACIFICI, G. 2007. A Scalable Application Placement Controller for Enterprise Data Centers. In *Proc. ACM WWW*.

URGAONKAR, B., SHENOY, P., AND ROSCOE, T. 2002. Resource Overbooking and Application Profiling in Shared Hosting Platforms. In *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

WANG, M., MENG, X., AND ZHANG, L. 2011. Consolidating Virtual Machines with Dynamic Bandwidth Demand in Data Centers. In *Proc. of IEEE INFOCOM 2011 Mini-Conference*.

XIE, D., DING, N., HU, Y. C., AND KOMPELLA, R. 2012. The Only Constant is Change: Incorporating Time-Varying Network Reservations in Data Centers. In *Proc. of ACM SIGCOMM*.

XU, H. AND LI, B. 2013. Joint request mapping and response routing for geo-distributed cloud services. In *INFOCOM, 2013 Proceedings IEEE*. IEEE, 854–862.

YU, B. AND PAN, J. 2015. Location-aware associated data placement for geo-distributed data-intensive applications. In *Proc. of IEEE Infocom 2015*.