Leveraging Video Viewing Patterns for Optimal Content Placement

K. -W. Hwang, D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, V. Misra, K. K. Ramakrishnan, D. F. Swayne

Leveraging Video Viewing Patterns for Optimal Content Placement

K.-W. Hwang[1], D. Applegate[2], A. Archer[2], V. Gopalakrishnan[2], S. Lee[2], V. Misra[1], K.K. Ramakrishnan[2], and D.F. Swayne[2]

[1]Columbia University, New York, NY, 10027 USA
[2]AT&T Labs Research, 180 Park Ave, Florham Park, NJ, 07932 USA

Abstract. As IP becomes the predominant choice for video delivery, storing the ever increasing number of videos for delivery will become a challenge. In this paper we focus on how to take advantage of user viewing patterns to place content in provider networks to reduce their storage and network utilization. We first characterize user viewing behavior using data collected from a nationally deployed Video-on-Demand service. We provide proof that users watch only a small portion of videos (not just for short clips, but even with full-length movies). We use this information and a highly flexible Mixed Integer Programming (MIP) formulation to solve the placement problem, in contrast to traditional popularity-based placement and caching strategy. We perform detailed simulations using real traces of user viewing sessions (including stream control operations such as Pause, Skip, etc.). Our results show that the use of a segment-based placement yields substantial savings both in storage as well as network bandwidth. For example, compared to a simple caching scheme using full videos, our MIP-based placement using segments can achieve up to 71% reduction in peak link bandwidth usage.

Keywords: Video-on-demand, content placement, user viewing pattern

1 Introduction

Video-on-Demand has grown rapidly to be the leading source of Internet traffic, accounting for about 49.2% of peak aggregate traffic (up from 29.5% in 2009) in North America [4]. To keep up with this rapid growth, providers have adopted a range of approaches including Content Distribution Networks (e.g., Akamai, Limelight), Peer-to-Peer based delivery (e.g., PPLive [1], PPStream, Zattoo) or hybrid combinations of the two (e.g., LiveSky [21]). Most of these approaches, however, focus on content delivery and ignore storage. Providers typically store a full copy of the content in multiple, if not all, locations. However, it has been known for a while that content popularity is skewed [10, 20, 22]; as a result not every video needs to have an equal number of copies in the network. More importantly, there is increasing evidence that users do not watch entire videos [20, 22]. Whether it is a short clip or a full movie, users stop watching the video before its end. Further, with the ability to skip or replay portions of the video, users tend to skip portions of the video they watch [9]. Taking advantage of this behavior to intelligently store content can significantly reduce storage cost and improve delivery efficiency.

Storage is cheap, so why bother? First, while the cost of consumer storage has fallen significantly (e.g., about US $100 for 2TB), carrier grade storage servers, as needed by a high performance and highly available streaming solution, are many orders more expensive for the same disk capacity [2]. Most of the cost comes from the strict redundancy requirements and the sustained streaming rate needed for video delivery. Second, to cater to growing demand and to gain competitive advantage, providers have been growing their library catalogues at a rapid rate [3]. Third, transferring this ever increasing number of videos to all the storage locations not only strains the distribution system but is also inefficient especially when not all of that data will be viewed. We therefore believe that storing only portions of the video at select locations results in worthwhile savings to providers. It reduces costs, improves efficiency, and opens opportunities to scale the library significantly for the same amount of disk space.

In this paper, we seek to leverage users' viewing patterns to design a more efficient video placement strategy. Ideally, we want to identify select portions for each video that can satisfy a large number of requests. By replicating those portions appropriately, we seek to maximize the number of requests satisfied locally with a minimum of disk usage. We first analyze video access data from a nationally deployed VoD service and confirm that many users indeed consume only a small fraction of each video they request (Section 4) even after factoring in parameters such as video size, popularity, and price. Based on this finding, a natural approach is to break a video into multiple segments, which become units for storing and fetching content.We study the advantages gained by two approaches: (a) splitting a video into a prefix and a suffix, and (b) splitting the video into fine-grained segments (e.g., chunks as in P2P systems).

Various studies have looked at the problem of content placement. For example, many approaches [7, 8, 16] use optimization-based techniques to guide placement. These, however, resort to heuristics to solve the problem or ignore important constraints such as link bandwidth. As a result, simple caches [5, 21] and cache replacement policies [5, 18] have been used within provider networks. Caching, however, can only be successful if the available cache is large enough to store the "working set". This working set is the set of unique videos requested over a time window. As we show in Section 4, the working set size at a location can be as large as 50% of the entire library. In this setting, smaller caches result in significant churn, rendering the cache ineffective.

Instead, we use a mixed integer program (MIP) formulation, whose solution guides how to place each segment at the different locations (Section 5). The MIP formulation takes a projected demand for each segment, a disk constraint at each location, and a bandwidth constraint for each link, and computes a solution that minimizes the overall network utilization. While the formulation and the solution technique draw on our previous work [6], the problem of placing segments adds new challenges. For example, with chunks, the scale of the problem grows so large that even the approach in our earlier work [6] is not able to solve the MIP quickly. As a result, we seek to cluster different chunks into groups and then place these chunk groups.

We use detailed simulation experiments and compare our optimal placement approach against alternate schemes (Section 6). Our results show that, there is tremendous benefit to storing just portions of a video. For example, just utilizing prefixes in the MIP-based placement results in up to 62% reduction in peak link bandwidth usage. Chunk-based MIP placement improves it further with a 71% reduction in peak link bandwidth usage over a cache of full-length videos using the least recently used (LRU) replacement policy. In fact, we see a 52% reduction even over a MIP-based placement of full-length videos.

2 Related Work

Existing work can be classified into three broad categories: studying user behavior with online videos, prefix caching, and optimal content placement.

User Behavior in Online Video Watching: With more deployments of IPTV and P2P-based VoD systems cropping up, various studies have analyzed different aspects of how users watch the videos. Yu et al. [22] study the user behavior in PowerLive, a VoD system deployed in China. Huang et al. [12] study different aspects the PPLive [1] P2P VoD system. Yin et al [20] study user behavior using the data collected from VoD access to the 2008 Beijing Olympics. Gopalakrishnan et al. [9] study how users use the DVD-like operations (e.g., skip, fast-forward, rewind) in a large-scale VoD system. Despite studying various important aspects of user behavior, none of these study how much of a video users watch. In this paper, we not only provide concrete evidence of that fact, but also study how user behavior correlates to different aspects such as video length and popularity.

Utilizing Video Prefixes: Existing work on video prefixes have focused on caching video prefixes, either for fast startup, or to reduce cache space. Sen et al. [15] and Zhang et al. [23] propose caching an initial prefix of video to provide fast start-up, deal with jitter, and reduce server load. Park et al. [13] propose a caching scheme that stores a portion of the video, where the size of the cached portion is determined by its popularity. Wang et al. [17] try to minimize aggregate network bandwidth by analytically determining the optimal proxy prefix cache. Wu et al. [19] propose partitioning videos into exponentially increasing segments and caching these segments. They also propose cache admission and replacement strategies that determine which segments to cache and replace. More recently, Huang et al. [11] and Allen et al. [5] study the use of P2P nodes for caching videos. The goal of both approaches is to reduce server load by using peers to serve the content. In contrast, our focus is to take advantage of client viewing patterns in placing content among the backend servers.

Optimal Placement Approaches: While content placement and replication has been a topic of extensive research, most existing work focuses on minimizing latency given constraints (e.g., server capacity [14]), but ignores network link capacity. More recent content placement schemes using an analytical framework include the work by Valancius et al. [16], which proposes an LP-based heuristic to identify which videos and how many replicas to be placed at customer home gateways. Borst et al. [8] focus on minimizing link bandwidth utilization assuming

a tree structure with limited depth. Both of these proposals focus on the delivery from the server to viewers; instead, we consider placing content among the servers in different locations. Baev et al. [7] consider the *data placement problem* where the objective is to minimize the cost without considering link bandwidth. They prove that the problem is NP-hard and that with different object lengths, no approximation algorithm is possible unless P=NP. Our problem is strictly more complex (due to link constraints) and also NP-hard since the data placement problem is a special case. We also find provably-good solutions (e.g., within 1% of optimal) on instances arising from real-world large-scale systems having diverse objects. Zhou and Xu [24] consider the problem of minimizing the load imbalance among servers subject to disk space and server network bandwidth constraints. Their work only considers the egress link capacity from servers.

3 Description of Data Set

We obtained trace data collected for 16 days in January, 2010 from a nationally deployed VoD service. This trace includes over 13 million requests for both free and paid videos belonging to various classes including music videos and trailers, TV shows, and full length movies. A record in this anonymized trace data includes information such as the requested video ID (typically a hash of the content), its length, its price, the time of request, and the location of request (i.e., city). It also contains information about the set of VCR-like operations that the user performed while watching the video in a given *session* (i.e., period from when the user started watching the video to when they stopped watching it).

Note that the session duration may be longer than the video length because the user may have re-wound or paused the video. Further, the actual amount of video watched is quite different from the session duration or even the video length. To get an accurate estimate of how much of the video was actually watched, we included the time spent in VCR-like operations. For example, suppose that a user spends the first 4 minutes fast-forwarding the video at 2x speed and then watches another minute before stopping the session. While the session length is 5 minutes, the amount of video consumed is 9 minutes. We measure the actual length of the video watched as a fraction of the video length and call it the *Normalized Length Viewed (NLV)*. In the above example, if the length of the video is 30 minutes, then NLV is 9/30=0.3.

Limitations of our data: The VoD service allows users to watch a video partially and continue at a later time. However, our data did not clearly identify if a particular session by a user was a new one or the resumption of a previous session. To overcome this, if a user requests the same video in two consecutive sessions that are less than T_r apart in time, then we assume that the later session resumes from where the previous session left off. Otherwise, we treat the later session as a new session starting afresh. As expected, we observed that the NLV initially grows larger with larger T_r. However, as we increase the threshold, the change in NLVs decreases (e.g., T_r=1hr and T_r=4hr result in almost identical NLVs). In the rest of the paper, we use T_r=4hr since the number of consecutive sessions beyond this threshold was negligible in our trace data.

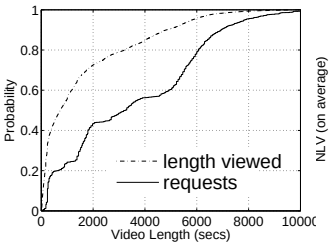


Fig. 1. Cumulative distribution of requests and viewing length of videos of different sizes.

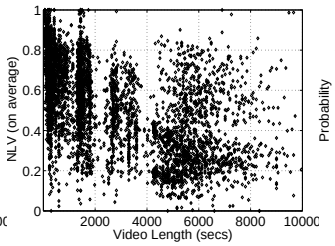


Fig. 2. Scatterplot of averaged NLV for each video of different lengths. One point per video.

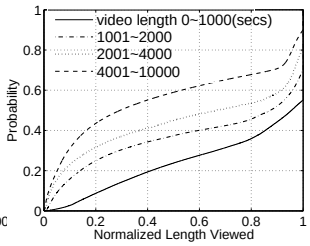


Fig. 3. Cumulative distribution of NLV for different classes of video lengths.

The VoD service provides free short previews for some of the paid content. Unfortunately, our trace data does not have information that allows us to distinguish which videos have previews and for how long. We analyzed our data and observed that viewings for most of the previews lasted less than 5 minutes. Since these preview sessions are short, not identifying access to these previews can significantly skew our results. In the rest of this paper, we exclude all sessions for these videos whose session length is shorter than 5 minutes. Note that both the heuristics, viz., session resumption and removing preview sessions, intentionally used conservative thresholds.

4 Motivation for Placing Video Segments

By analyzing our data described in the previous section, we now provide the motivation for placing segments rather than full videos. We study how much of a video users watch, and then investigate how this correlates to video properties such as video length and popularity.

4.1 Most Users Watch Fraction of a Video

We first study the relation between the length of requested videos and the NLV for these viewing sessions. In Figure 1, we show the video length and the viewed length for all requests and plot the cumulative distribution. We observe that 43% of the requests are to videos that are 2000 secs (e.g., a 30 minute TV show) or less. By contrast, the amount of video watched by users is much shorter. Specifically, about 73% of the cases watched 2000 seconds or less of the requested video. This result indicates that most users do not watch videos fully.

4.2 Correlation between NLV and Video Properties

Having established that viewers watch only a portion of the video, we examine the correlation between the length of a video and the NLV. In Figure 2, we show a scatter plot of the average NLVs for all requests to each video of different lengths. We see that shorter videos tend to have larger NLVs. We also see four clusters of video lengths. This is not surprising given the nature of the content in the library: music videos, trailers, episodes of shows, documentaries, and movies.

In Figure 3, we plot the cumulative distributions of the four clusters of video lengths as identified in the previous figure. We clearly see that *longer videos*

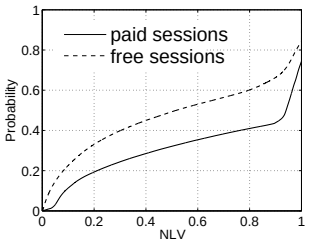


Fig. 4. Cumulative distribution of the NLVs for free and priced videos.

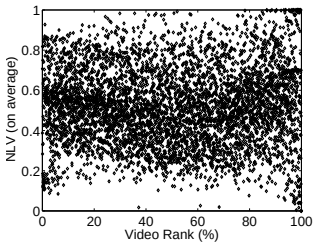


Fig. 5. NLV for videos of decreasing popularity. (100%: the least popular)

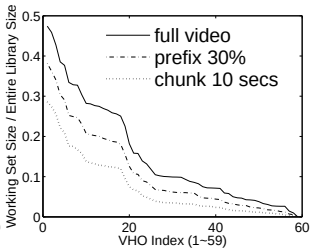


Fig. 6. Fraction of working set size to the entire library size at each VHO.

tend to have smaller NLV values. For the cluster of videos of the greatest length, around 55% of the requests stopped before reaching 40% of the video length. By contrast, for the group of shortest videos, only less than 20% of the requests stopped before viewing 40% of the video.

We separately explore sessions for free and paid videos since it reasonable to expect viewers to watch longer when they pay for the video. Figure 4 shows that *paid videos have longer NLV than free videos*, confirming that widely held belief. For example, about 45% of the requests for free videos stopped watching before 40% of the video while only 25% for paid videos stopped at that point.

We studied the relationship between the popularity of a video and how much of it is viewed. Conventional wisdom says that popular videos tend to be viewed fully. This belief also forms the basis for some of the prefix caching approaches [13]. For each video, we calculated the average NLV across all requests to the video and plot the result in Figure 5. Surprisingly, the figure shows that there is *little to no correlation between the popularity of a video and its NLV*. This result indicates that just taking the popularity of a video and deciding how to store it may lead to an inefficient placement decision and that there are benefits to be gained by looking to store videos at a finer granularity.

Finally, we also studied the correlation between NLV and other properties (e.g., size, genres, weekdays vs. weekends). We observe *no strong correlation* between NLV and those properties. We do not present the detailed findings here due to the space limit.

4.3 Working Set Size

The working set in a city's video hub office (VHO) is defined as the number of unique videos that are requested at that office within a time window. The working set is important as it determines the effectiveness of caching. There will be a large amount of cache cycling if the working set size is larger than the cache size. In Figure 6, we plot the working set size (in terms of bytes compared to the library) at 59 VHOs during the peak hour on a Saturday. We consider three cases: (1) full videos, (2) prefixes and suffixes, and (3) 10-second chunks. For the prefix/suffix case, we set the prefix to the initial 30% of a video (see Section 6.5).

We observe that the working set size at a VHO can be as high as 48% of the library size. The top 20 VHOs have a working set of 20% or more. With prefixes, the relative working set size can be as large as 39%, and the top 20 VHOs have

a working set size of 15% or larger. Using chunks reduces the relative working set size, but is still quite large (as high as 30%). This result shows that cache space at each VHO has to be quite large for caching to be successful.

5 MIP Formulation

In this section we briefly present the mixed integer program (MIP) we use to determine the optimal content placement. While we have presented this formulation in our earlier work [6], we describe it here for completeness. We also discuss new aspects of the problem when we consider placing segments of a video.

5.1 Problem Formulation

$$\min \sum_{m \in M} \sum_{i,j \in V} s^m a_j^m c_{ij} x_{ij}^m \tag{1}$$

$$\text{s.t.} \sum_{i \in V} x_{ij}^m = 1, \ \forall m \in M, j \in V \tag{2}$$

$$x_{ij}^m \leq y_i^m, \ \forall i, j \in V, m \in M \tag{3}$$

$$\sum_{m \in M} s^m y_i^m \leq D_i, \ \forall i \in V \tag{4}$$

$$\sum_{m \in M} \sum_{\substack{i,j \in V: \\ l \in P_{ij}}} r^m f_j^m(t) x_{ij}^m \leq B_l, \ \forall l \in L, t \in T \tag{5}$$

$$x_{ij}^m \geq 0, \ \forall i, j \in V, m \in M \tag{6}$$

$$y_i^m \in \{0, 1\}, \ \forall i \in V, m \in M \tag{7}$$

Let V denote the set of VHO locations, L the set of directed links between these locations, W the set of videos in our catalog, and M the set of distinct segments (e.g., chunks, or prefix and suffix) of videos. (In case a video is not broken into multiple segments, $M = W$.) The set of time slices at which we enforce the link bandwidth constraints is T. Each VHO i has disk capacity D_i, and the size of segment $m \in M$ is s^m. For each pair of VHOs $i, j \in V$, we assume a fixed directed path P_{ij} from i to j, where $P_{ii} = \emptyset$. The bandwidth of link $l \in L$ is B_l, while the bit rate of video $m \in M$ is r^m (both in Mbps). For each video $m \in M$, VHO $j \in V$ receives a_j^m requests during the entire modeling period. Given a time slice $t \in T$, the number of concurrent streams is $f_j^m(t)$. c_{ij} denotes the cost of serving one GB of video from i to j. Our MIP model has two types of decision variables. For each $i \in V$ and each $m \in M$, y_i^m indicates whether we store m at i. y_i^m is a binary variable, because given m, i always stores it in its entirety or not. When a request for m arrives at VHO j, the variable x_{ij}^m tells what fraction of requests should be served from VHO i. The objective expressed by (1) is to minimize the overall cost of byte transfer while serving *all* the requests for the entire period (Constraint (2)) without violating the disk capacity and link bandwidth capacity constraints (Constraints (4) and (5)). Constraint (3) captures the fact that location i can serve m only when it has a local copy. Note that Constraints (2) and (3) combine to ensure that every $m \in M$ is available in the system, i.e., $\sum_{i \in V} y_i^m \geq 1, \ \forall m \in M$.

5.2 Managing the Scale of the MIP

Finding an optimal solution for the MIP outlined above is NP-hard. We have presented a practical approach to solve the MIP quickly in our earlier work [6]. However, despite the advances presented in that work, the scale of the problem grows significantly when we attempt to place chunks. For instance, with 1-hour videos, breaking the videos into 10-second chunks increases the number of variables in the MIP formulation by a factor of 360.

To make the problem tractable even with small chunks, we *cluster* chunks into groups. We then solve the MIP instance based on these groups and place the groups as dictated by the output of the MIP. We first considered creating clusters by binning chunks of same popularity. This approach, however, has an important drawback in that it results in unbalanced cluster sizes. In particular, clusters with unpopular chunks are extremely large. For example, with our data, we noticed that the size of the largest cluster was ∼130000 times that of the smallest cluster. The size of these large clusters makes it very hard to place them in any VHO, thereby defeating the very purpose of MIP-based placement.

To address this, we sorted all the chunks in the decreasing order of number of accesses. Then, we clustered groups of n chunks: the first cluster had the n most popular chunks, the second cluster had the next n popular chunks, and so on. The downside of this approach is that it can potentially result in more copies of certain chunks than needed (because of the additive effect of the demand of each chunk). However, by limiting the size of a cluster to a small number, we can minimize this effect while still solving the MIP quickly.

6 Experimental Results

We evaluated the performance of the chunk and prefix-based placement strategies through simulation experiments using data from a nationally deployed VoD service. We describe our experiment setup and present the results in this section.

6.1 Experiment Setup

We use a custom event-driven simulator to perform trace-driven simulations. We use a network modeled after a deployed IPTV network. This network has 59 nodes (i.e., VHOs) with 70+ bi-directional links of equal bandwidth connecting the different VHOs. We assume that all the VHOs have the same amount of storage. However, we vary this size in different experiments to understand the trade-offs between disk and link bandwidth capacity.

To protect proprietary information, we use normalized disk sizes relative to the space needed to store the entire video library. For instance, when the total disk space across all VHOs is 2x the space needed to store the library, each VHO has disk space equal to ∼3% (2/59) of the total library space. We partition the disk at each VHO into two: One part stores the videos assigned to the VHO (i.e., not eligible for replacement), and the other is used as a dynamic cache (using LRU replacement). We also use a small cache equal to 5% of the space at each VHO (i.e., 5% of 3%) with our MIP-based placement schemes to accommodate for errors in demand estimation and to handle flash crowds. When the video

requested by a user is available locally (either in the pre-assigned portion or in the cache), the VHO simply handles the request locally. If not, the VHO fetches the video (or the segment of the video) from a remote VHO that stores it.

We assume that the bitrate of videos is 2 Mbps. We use 10-second chunks to take advantage of skip operations. Recall that we cluster chunks to solve the MIP quickly. We use cluster sizes of 100 chunks as this value gave us a good balance between the time taken to solve the MIP and the approximation introduced in the placement solution. Similarly for prefixes, we experimented with various prefix sizes and determined that a prefix size of 30% gave the best results.

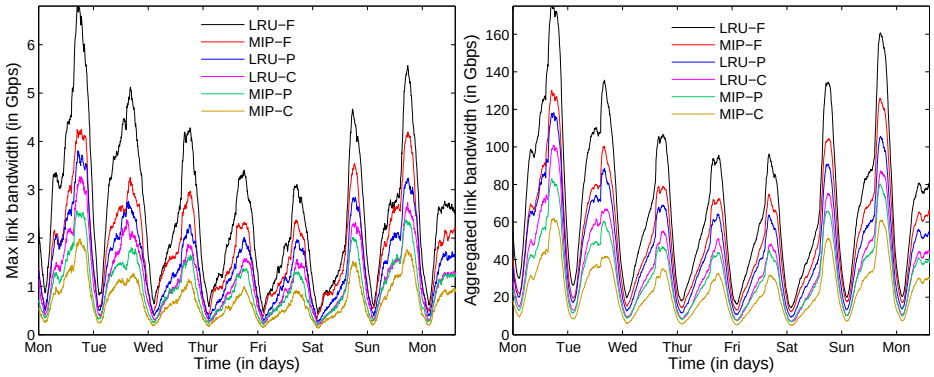
We use the data trace described in Section 4 to simulate video requests by users. We translate the stream control operations in the data (e.g., skip, fast-forward) to accesses to specific segments of the video. When a user performs a skip, we treat the corresponding portion of the video as *not* accessed. However, we consider that all the segments are accessed during a fast-forward operation because the video is displayed during the operation albeit at a faster rate. Note that we *do not* stop a remote transfer, from a different VHO, prematurely even if the local user abandons that video. This allows us to cache the entire video segment and serve it locally when it is requested again.

We are primarily interested in the amount of data stored and transferred among the VHOs; we do not focus on the transfer from a VHO to the end user. However, by taking available capacity into account as part of the MIP, we ensure that the solution *guarantees* sufficient capacity to serve each user request in time, so that users experience no interruptions.

Comparison with LRU-based Caching: Previous work showed that LRU-based caching outperforms popularity based placement [6, 18]. As a result, we compare the performance of our placement scheme with LRU-based caches. To have an apples-apples comparison with the MIP-based placement, we assume that each VHO has a disk space equal to 3% of the library size. For LRU-cache based approach, we place one copy of every video at a random location and allow the dynamics of the requests to cache copies at the other VHOs. On average, a VHO uses around 50% of disk space for the dynamic cache. We also use 10-second chunks and 30% prefixes when experimenting with chunks and prefixes respectively. With the LRU-cache, we assume an idealized setting where each VHO knows the nearest location with a copy of the requested object.

6.2 Maximum and Aggregate Link Bandwidth

We quantify the benefit of placing segments of the video instead of the full video by comparing the peak and aggregate bandwidth needed for each of the approaches. Recall that our data was spread over 16 days. We use the first 9 days of data to predict the demand for videos (or segments) with the MIP-based placement, and to warm up the caches with LRU. At the end of 9 days, we solve the MIP formulation and place content accordingly. We then simulate for the last 7 days and measure the performance of both LRU and MIP-based placement. In [6] we addressed in detail the practical considerations for the algorithm design and parameter selection in demand estimation, time-varying demand, and placement update frequency.



(a) Maximum link bandwidth (b) Aggregate link bandwidth

Fig. 7. Link bandwidth utilized across all links, computed every 5 minutes (the ordering of the curves is the same as in the legend).

Figure 7(a) shows the maximum link bandwidth used across all links in the network measured in 5-minute windows over the last 7 days for full videos (MIP-F, LRU-F), 30% prefixes (MIP-P, LRU-P), and 10-second chunks (MIP-C, LRU-C). For the MIP-based schemes the peak bandwidth needed goes down from about 4.2 Gbps with MIP-F to 2.6 Gbps with MIP-P (a 38% reduction). MIP-C performs the best with a peak of only 2.0 Gbps. In Figure 7(b), we plot the total bytes transferred across all links in the network, averaged over 5-minute windows. MIP-F results in a maximum of 130 Gbps of aggregate traffic in the network while MIP-P has a maximum of 83 Gbps (36% reduction). At 61 Gbps, MIP-C again results in the least amount of data transferred.

Furthermore, we see that the MIP-based placement outperforms LRU-based caches. For example, MIP-F requires a maximum of 4.2 Gbps bandwidth, while LRU caching of full videos (LRU-F) requires 6.9 Gbps. We observe similar trends when we compare MIP-P and MIP-C with corresponding segment-based caching schemes LRU-P and LRU-C, respectively (e.g., 33% reduction from LRU-P to MIP-P). More interestingly, we see that LRU-C requires *more* bandwidth than even MIP-P. Between the worst case (LRU-F) and the best case (MIP-C), we achieve over a factor of 3 improvement in max. and aggregate network bandwidth.

These two results confirm our hypothesis: *Taking advantage of user behavior and placing segments provides significant benefit.* By using prefixes and suffixes, we take advantage of the fact that people do not watch full videos. We get further benefits when using chunks because of two factors: (a) using chunks, we need not transfer unwatched portions of even prefixes or suffixes when users abandon a video, and (b) we can take advantage of fine-grained stream control such as skip operations to eliminate data transfer within a prefix or suffix.

6.3 Number of Replicas and Hop Count

We examine the number of copies for each *object* across all the VHOs, where an object is a full video in MIP-F or a segment in MIP-P and MIP-C. In Figure 8 we

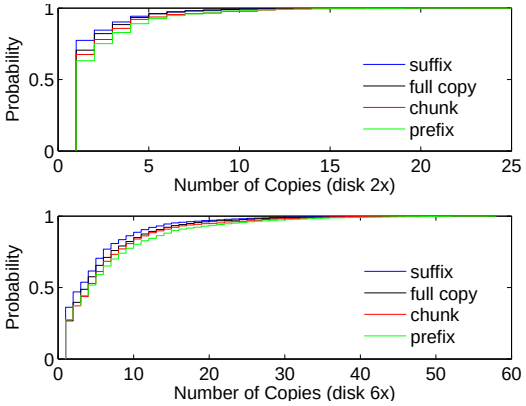


Fig. 8. Cumulative distribution of the number of replicas when the available disk is 2x and 6x of library size (the ordering of the curves is the same as in the legend).

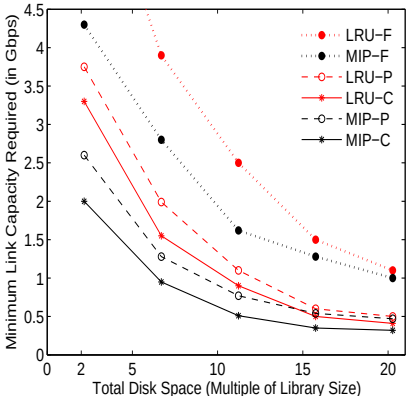


Fig. 9. Feasibility region of Placement-based and Caching-based solutions (the ordering of the curves is the same as in the legend).

show the cumulative distribution when the aggregate disk space across all VHOs is 2x and 6x the library size. When the total disk space is only 2x the library size, even the most popular object is not replicated in all 59 VHOs, and more than 60% of all objects have only one replica. When the disk space increases to 6x the library size, we see that the placement solution takes advantage of this extra space and places a copy of the popular objects in every location. We also see a greater spread in the number of copies of each object with 50% of the objects having anywhere from 2 to 15 replicas. We observe that prefixes tend to be copied in more VHOs than suffixes or full videos. For instance, with the disk space of 2x, there is only one copy for 63% of prefixes, 71% of full videos, and 78% of suffixes. On average, there are 2.2 copies for a prefix, 1.7 for a suffix, and 1.8 for a full video. We also see that by flexibly placing fine-grained chunks, MIP-C replicates medium-popularity videos in more locations.

This result is not surprising since splitting videos allows us to "pack" it better into the available space. However, this result shows an important aspect of VoD service and the need for more intelligent placement strategies such as our MIP based scheme: with videos, despite the skew in popularity, there is a large set that is requested enough number of times to not be ignored. Existing schemes (e.g., replicating top K videos in all locations) ignore this aspect and hence are unlikely to result in as good performance, as shown in [6].

Hops Traversed: We measured the effectiveness of the placement in terms of the number of hops traversed to fetch videos from remote VHOs. Due to space limitations, we only briefly summarize the results. Our results show that all MIP-based schemes show a similar hop count distribution. When the aggregate disk space is 2x the library size, the average is about 2.1 hops, with less than 10% of transfers taking more than 4 hops. In contrast, the caching-based approaches have more than 20% of the transfers longer than 4 hops resulting in a larger

Prefix Size	10%	20%	30%	40%	50%	100%
LRU	1689	1497	1451	1496	1565	2203
MIP	1425	1274	1220	1254	1284	1832

Table 1. Total bytes transferred from remote locations for different prefix sizes (in TB)

	full video	prefix 30%	chunk 10sec	chunk 1min
total	7483	4875	3563	3664

Table 2. Total served bytes for different segment sizes (in TB)

average hop count (2.8 hops). This, along with the result in Figure 8, shows that MIP-based solutions do an effective job of placing objects in the right locations.

6.4 Disk and Bandwidth Tradeoff

We study the tradeoff between disk and link bandwidth capacity by varying the aggregate VHO disk size between 2x the library size and 20x and measuring the minimum bandwidth required between each VHO. As expected, we see in Figure 9 that the amount of bandwidth required decreases in all cases as the total disk space increases. However, by taking user behavior into account, segment-based approaches (MIP-P, MIP-C, LRU-P, and LRU-C) consistently require less bandwidth than storing full videos (MIP-F and LRU-F). This is true even when the disk space in the system is 20x the library size, where each VHO has enough space to store around one third of the entire library. We also observe that the MIP-based solutions outperform caching-based solutions, especially when the disk space is scarce. This shows the importance of having storage space having at least the "working set size" with caching schemes.

6.5 Effect of Segment Size

We compare the effects of different prefix sizes in Table 1 by comparing the total bytes transferred from remote VHOs for MIP-P and LRU-P. As the table shows, the total bytes transferred initially decreases with increasing prefix size, but eventually starts increasing. The least amount of data is transferred when the prefix size is 30%. We also experimented with per-video prefix values. However, our results showed very modest improvements compared to a fixed prefix size.

To understand the effect of chunk size, we calculate the total bytes the system serves both locally and remotely ($\sum_{j,m} a_j^m s^m$ using the notation in Section 5) in Table 2. We observe that with 10-second chunks, the system needs to serve only 48% of bytes served when using full videos (3563 vs. 7483 TB). We also find that the amount of savings due to skip operations is 101TB for the 10-second chunks (about 3% of the total reduction of 3920TB) and that the difference between 1-minute chunks and 10-second chunks is only modest. This result indicates that the majority of gain with the chunk-based placement comes from being able to stop video transfer shortly after a user aborts a session. We speculate that this is because of limitations of existing VoD interfaces and that this may change when DVD-like navigation becomes possible.

6.6 Cache Dynamics

All these approaches make use of caches; MIP uses small caches and LRU caching uses large ones. Hence, we looked at how placement affects the performance of

these caches. We study the number of remote transfers (equivalent to cache misses), the number of cache replacements, and the number of failed cache insertions. Instead of counting the number of occurrences, we study these factors in terms of TB of data to account for the heterogeneity in object sizes.

	LRU-F	LRU-P	LRU-C	MIP-F	MIP-P	MIP-C
Remote Transfer	2115	1451	1147	1839	1220	921
Cache Replacement	177	365	1104	34	109	918
Failed Insertions	1841	1081	0	1805	1107	0

Table 3. Cache dynamics for the different schemes over the last 7 days (in TB).

As expected, we see in Table 3 that the MIP-based, segment placement schemes result in the least amount of data transferred. More interesting are the results of cache replacements and failed insertions. A failed insertion occurs when there is insufficient space in the cache and none of the existing objects can be replaced because they are all being used. It is of particular significance with videos because it means that all the work done in transferring this large object will have to be repeated again. Table 3 shows that despite having 1/10th the cache size, MIP-P experiences 70% less replacements and approximately the same amount of failed insertions compared to LRU-P. This shows that the cache in MIP-P is being used effectively; objects in the cache are used and hence cannot be replaced. MIP-C and LRU-C do not have failed insertions because all objects are small and of the same size. However, this also results in a lot of cache churn.

7 Conclusions

We proposed a new approach for placement of on-demand videos in a backbone network. We took advantage of the fact that users (a) do not watch videos fully, and (b) skip over portions of the video. Our placement approach is based on a Mixed Integer Program that places segments of each video (chunks, or prefixes and suffixes) across locations in the backbone. Using traces of user viewing behavior from a nationally deployed VoD service, we show that our MIP-based placement significantly outperforms simple schemes like LRU caching. Importantly, the MIP-based placement consistently resulted in lower bandwidth usage.

Our results show that the bulk of the improvement comes from users abandoning videos, with additional, albeit small, benefit coming from users skipping portions of the video. This suggests that despite the availability of stream control functions like skip, users view large parts of a video sequentially. It would be interesting to understand how this behavior would change as VoD providers support more flexible DVD-like navigational capabilities in the future, and its impact on content placement.

References

1. Pplive, `http://www.pplive.com`

2. Sun storage 6180 array, http://www.oracle.com/us/products/servers-storage/storage/disk-storage/047193.html
3. Comcast VoD Choices Hits 25K (May 2010), http://www.lightreading.com/document.asp?doc_id=191777&site=lr_cable
4. Sandvine global internet phenomena report (May 2011), http://www.sandvine.com/news/pr_detail.asp?ID=312
5. Allen, M., Zhao, B., Wolski, R.: Deploying video-on-demand services on cable networks. In: Proceedings of IEEE ICDCS. Toronto, Canada (June 2007)
6. Applegate, D., Archer, A., Gopalakrishnan, V., Lee, S., Ramakrishnan, K.K.: Optimal content placement for a large-scale vod system. CoNEXT, ACM (2010)
7. Baev, I.D., Rajaraman, R., Swamy, C.: Approximation algorithms for data placement problems. SIAM J. Computing 38(4), 1411–1429 (2008)
8. Borst, S., Gupta, V., Walid, A.: Distributed caching algorithms for content distribution networks. In: Proceeding of IEEE INFOCOM (2010)
9. Gopalakrishnan, V., Jana, R., Ramakrishnan, K.K., Swayne, D.F., Vaishampayan, V.A.: Understanding couch potatoes: Modeling interactive usage of iptv at large scale. In: Proceedings of ACM IMC (2011)
10. Guo, L., Tan, E., Chen, S., Xiao, Z., Zhang, X.: Does internet media traffic really follow zipf-like distribution? In: Proceedings of ACM SIGMETRICS (2007)
11. Huang, C., Li, J., Ross, K.W.: Can internet video-on-demand be profitable? In: Proceedings of ACM Sigcomm (2007)
12. Huang, Y., Fu, T.Z.J., ming Chiu, D., Lui, J.C.S., Huang, C.: Challenges, design and analysis of a large-scale p2p vod system. In: ACM SIGCOMM (2008)
13. Park, S.H., Lim, E.J., Chung, K.D.: Popularity-based partial caching for vod systems using a proxy server. In: Proceedings of IEEE International Parallel and Distributed Processing Symposium (2001)
14. Qiu, L., Padmanabhan, V.N., Voelker, G.M.: On the placement of web server replicas. In: Proceeding of IEEE INFOCOM (2001)
15. Sen, S., Rexford, J., Towsley, D.: Proxy prefix caching for multimedia streams. In: Proceedings of IEEE INFOCOM (1999)
16. Valancius, V., Laoutaris, N., Massoulié, L., Diot, C., Rodriguez, P.: Greening the internet with nano data centers. In: Proceedings of ACM CoNEXT (2009)
17. Wang, B., Sen, S., Adler, M., Towsley, D.: Optimal proxy cache allocation for efficient streaming media distribution. In: Proceedings of IEEE INFOCOM (2002)
18. Wu, J., Li, B.: Keep cache replacement simple in peer-assisted vod systems. In: Proceedings of IEEE INFOCOM. pp. 2591–2595 (April 2009)
19. Wu, K.L., Yu, P.S., Wolf, J.L.: Segment-based proxy caching of multimedia streams. In: World Wide Web. pp. 36–44. Hong Kong (2001)
20. Yin, H., Liu, X., Qiu, F., Xia, N., Lin, C., Zhang, H., Sekar, V., Min, G.: Inside the bird's nest: measurements of large-scale live VoD from the 2008 olympics. In: Proceedings of the ACM SIGCOMM Internet measurement conference (2009)
21. Yin, H., Liu, X., Zhan, T., Sekar, V., Qiu, F., Lin, C., Zhang, H., Li, B.: Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky. In: Proceedings of ACM Multimedia. pp. 25–34 (2009)
22. Yu, H., Zheng, D., Zhao, B.Y., Zheng, W.: Understanding user behavior in large-scale video-on-demand systems. In: Proceedings ACM SIGOPS/EuroSys (2006)
23. Zhang, Z.L., Wang, Y., Du, D.H.C., Su, D.: Video Staging: A Proxy-Server-Based Approach to End-to-End Video Delivery over Wide-Area Networks. IEEE/ACM Transactions on Networking 8(4) (August 2000)
24. Zhou, X., Xu, C.Z.: Optimal video replication and placement on a cluster of video-on-demand servers. In: Proc. IEEE ICPP (2002)