Cache-Centric Video Recommendation: An Approach to Improve the Efficiency of YouTube Caches

DILIP KUMAR KRISHNAPPA and MICHAEL ZINK, University of Massachusetts Amherst
CARSTEN GRIWODZ and PÅL HALVORSEN, Simula Laboratory, University of Oslo, Norway

In this article, we take advantage of the user behavior of requesting videos from the top of the related list provided by YouTube to improve the performance of YouTube caches. We recommend that local caches reorder the related lists associated with YouTube videos, presenting the cached content above noncached content. We argue that the likelihood that viewers select content from the top of the related list is higher than selection from the bottom, and pushing contents already in the cache to the top of the related list would increase the likelihood of choosing cached content. To verify that the position on the list really is the selection criterion more dominant than the content itself, we conduct a user study with 40 YouTube-using volunteers who were presented with random related lists in their everyday YouTube use. After confirming our assumption, we analyze the benefits of our approach by an investigation that is based on two traces collected from a university campus. Our analysis shows that the proposed reordering approach for related lists would lead to a 2 to 5 times increase in cache hit rate compared to an approach without reordering the related list. This increase in hit rate would lead to reduction in server load and backend bandwidth usage, which in turn reduces the latency in streaming the video requested by the viewer and has the potential to improve the overall performance of YouTube's content distribution system. An analysis of YouTube's recommendation system reveals that related lists are created from a small pool of videos, which increases the potential for caching content from related lists and reordering based on the content in the cache.

Categories and Subject Descriptors: H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information System—*Video*

General Terms: Measurement, Performance

Additional Key Words and Phrases: Caching, YouTube, recommendation

ACM Reference Format:
Dilip Kumar Krishnappa, Michael Zink, Carsten Griwodz, and Pål Halvorsen. 2015. Cache-centric video recommendation: An approach to improve the efficiency of YouTube caches. ACM Trans. Multimedia Comput. Commun. Appl. 11, 4, Article 48 (April 2015), 20 pages.
DOI: http://dx.doi.org/10.1145/2716310

1. INTRODUCTION

YouTube is the world's most popular Internet service that hosts user-generated videos. According to You Tube statistics [You Tube 2012], each minute 100 hours of new videos are uploaded to YouTube, viewers can choose from hundreds of millions of videos and over 6 billion hours of videos are watched each month on YouTube. Consequently, these numbers lead to a huge amount of network traffic, and Google (the owner of YouTube) maintains a substantial infrastructure to provide a reliable and well-performing video streaming service. For example, according to the GoogleTransparencyReport [Google 2012], on January 23 2014 at 10PM, the United States YouTube traffic made up more than 68% of the global Internet traffic. Google's approach to tackle these challenges is

Author's address: D. K. Krishnappa; email: dilip.manu@gmail.com.

DOI: http://dx.doi.org/10.1145/2716310

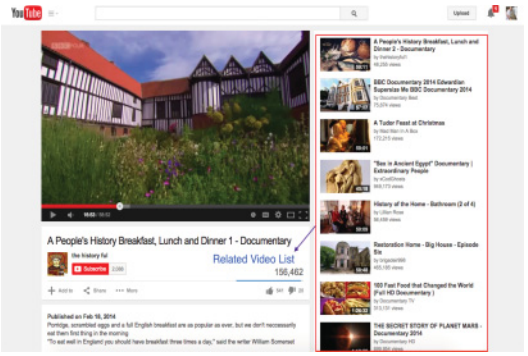


Fig. 1. This screenshot shows a YouTube video page and the location of the related video list on that page.

a network of caches that are globally distributed. With the aid of the caches, latency on the viewer's side and overall network utilization can be reduced.

Unfortunately, effective caching is much harder in the case of YouTube than in other video streaming services like Netflix or Hulu. This is caused by several facts. First of all, services that offer professionally produced content like movies and TV shows provide an online library on the order of several 10,000s of titles, a number that is much lower compared to the hundreds of millions of videos offered on YouTube. Second, the providers of purely professionally produced content determine when new content will be made available and, thus, can schedule the distribution of content into caches much better.

YouTube's dilemma, in terms of caching, is the fact that the popularity distribution of videos is a long-tail distribution. For the large number of videos in YouTube, this means that every cache that is limited to storing a small portion of the total number of videos will essentially perceive accesses to the tail as random choices, impossible to distinguish from a uniform distribution. This characteristic has already been shown in earlier studies. For example, in our trace-based characterization of YouTube traffic in a campus network [Zink et al. 2009], 70% or more of the overall videos are only requested once within 24 hours. In Cha et al. [2007], similar results are obtained by a global analysis of metadata made available by YouTube. This characteristic leads to the fact that many of the requested videos will not be available in a cache. This problem has been officially recognized by YouTube in a keynote address at MMSYS 2012, [YouTube 2012a], and the current approach is to increase the number of caches that serve content from the tail and fill these caches through prefetching.

In this article, we propose a different proxying approach that makes use of YouTube's recommendation system to increase the efficiency of its caches, since it is impossible to predict what a viewer is most likely to watch next. The general idea is to influence the video selection behavior of a viewer based on the content of the cache that serves the viewer. Each YouTube video web page offers a list of recommended videos, next to the video a viewer has selected (see Figure 1). In YouTube's own terms, this is described as the *related video list*. In our work, we exploit the fact that viewers frequently choose videos from the related list and present an approach in which the cache modifies the related video list that is sent from the server through the cache to the client. Our idea is to reorder the related list by moving the related list videos already present in the cache to the top of the list and move the rest to the bottom of the list based on stable sort [Knuth 1998].

To investigate the feasibility of our related list reordering approach, we perform an analysis on the chains and loops created by user behavior in selecting YouTube videos from the related list. Our results show that YouTube creates its related list from a

small pool of videos which provides potential for related list caching and reordering based on the content in the cache. In addition, we perform a measurement study from the University of Massachusetts Amherst campus network to determine the origin of videos that are requested from the related list. The results of this study indicate that YouTube uses a three-tier cache hierarchy to serve videos and that the origin is chosen in a way that achieves load balancing between the different levels in the caching hierarchy. Preference to the top videos of the related list for a requested video is not given.

In Krishnappa et al. [2013b], we present an hypothesis which suggests that users select from the top of the related list even when provided with a reordered related list. In order to verify our hypothesis, we perform a study with 40 actual users in which we present users a randomly reordered list and track the video they choose from that list. The results from our study strongly indicate that YouTube users prefer to select the next video they want to watch from the top of the related list even when provided with a reordered list.

To evaluate the proposed related list reordering approach, we perform a study based on YouTube traces obtained from the University of Connecticut campus network. Once provided with the reordered list, the user has two choices of video selection: (i) content centric selection, where users select the same content as they would before the reordering of related list; and (ii) position centric selection, where users select the next video from the same position in the related list after reordering. Based on these traces, we show that first, users indeed choose top ranked videos from the related video list, and second, with our proposed reordering approach and position centric selection, we can obtain a 2 to 5 times increase in cache hit rate. This increase in cache hit rate due to reordering of related list reduces the bandwidth consumption on the uplink from the level 1 cache to higher level caches and reduces the latency in serving the videos requested to the user.

The remainder of the article is structured as follows. Section 2 presents the related work in the area of caching YouTube related videos. In Section 3, we present the data sets we use to analyze our new caching approach, and analyze the potential influence of the related list on the viewers' video selection. Section 4 presents the related list differences based on global analysis and suggests the position of related videos to cache for maximum efficiency. Section 5 presents our cache-centric video recommendation approach and evaluates that approach with the aid of trace-based simulations. In addition, we also present the user study results of reordering related list randomly. A discussion of our results is presented in Section 6. Section 7 concludes the article.

2. RELATED WORK

The use of proxies and caches has been intensively studied in related work. In the following, we mention the ones closest to our work.

Podlipnig and Böszörmenyi [2003] provide an in-depth survey of various web cache replacement strategies and their advantages and disadvantages. Papadakis et al. [2013] identify the parameters and conditions that affect performance of content caching solutions and provide different caching schemes and algorithms to improve content caching capabilities. In Cha et al. [2007] and Zink et al. [2009], trace-driven simulations were performed to investigate the effectiveness of caching for YouTube videos. Although the traces for both studies were different, the results showed that caching can reduce server and network load significantly. Neither consider reordering the related video list to increase the efficiency of the cache.

Besides caching, YouTube's recommendation system (the related video list) has also been studied. Davidson et al. [2010] present an approach of the formulation of the YouTube recommendation system based on various factors and rankings. Zhou et al.

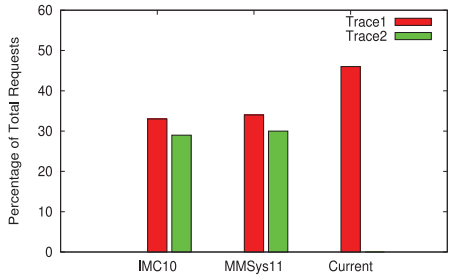


Fig. 2. Percentage of videos selected from related video list as reported in earlier studies (IMC10 [Zhou et al. 2010], MMSys11 [Khemmarat et al. 2011]) and obtained from an analysis of the trace used in this article.

[2010] and Khemmarat et al. [2011] have shown that users make significant use of the related video list. This means that, after watching the initially selected video, the viewer chooses to watch a video that is offered on the related video list next. Figure 2 shows a comparison of these result that were derived from five different datasets. It is important to mention that the results from Zhou et al. [2010] are obtained from two datasets of global meta information (taken at different points in time) that is offered by YouTube for each individual video. The results for Khemmarat et al. [2011] are obtained from two network traces taken at a campus gateway. From these results, we conjecture that both, on a global and regional level, ∼30% or more of video requests are made by the viewer by selecting a video from the related video list.

In Zhou et al. [2011], the authors perform further analysis of YouTube's recommendation system based on global statistics for YouTube videos. The authors show that the number of repeated selections from the related list is proportional to the position of the video in the related list (the higher the position of the video in the list, the higher the chance that it will be selected by the viewer).

As in Zhou et al. [2010], the results confirm our assumption on related list usage and support our related list reordering approach based on position of the video on related list. While the goal of the work presented in Khemmarat et al. [2011] was to show how the prefetching of prefixes from videos on YouTube's related list can improve caching, it also shows that the related list is often used by viewers to select a video (see Figure 2). In contrast to the work we present in this article, no modification of the related list at the cache is proposed.

Cheng et al. [2009] measured the YouTube video graph created by related video links and found that the graph has a large clustering coefficient and exhibits the small world property. A simulation-based evaluation of a P2P video sharing systems showed that if users use the related video list to browse videos, the percentage of source peers that have the requested video in their cache is high. Cheng and Liu [2009] also proposed a P2P video sharing system to reduce YouTube server load and suggested using prefetching based on YouTube's related video list at the clients of a P2P system to provide smooth transition between videos. Their evaluation was based on emulated user browsing pattern. The evaluation of their approach showed that it performs significantly better (55% hit ratio) comparing with a random prefetching approach (nearly 0% hitrate).

Adhikari et al. [2011, 2012] uncover the overall architecture of the YouTube video distribution system. While our results presented in Section 3.3 agree with their findings, their work does not investigate which of the videos of the related list are transmitted from which cache level. In addition, their work is not concerned with improving the efficiency of YouTube caches.

Chakareski [2011] takes a more general view on recommender systems (called catalogues in his case) to develop an optimization framework that has the goal to reward providers if an item selected by a customer can be provided immediately and penalize them if the provision is delayed. Similar to our case, the author shows that optimizing for immediate access to items at the beginning (or top) of a catalogue is beneficial and optimizes the reward for the provider. The article does not reveal how this immediate access can be provided, and we see our work as complementary since it addresses content provision. In addition, our study is based on actual user behavior.

Yoo and Kim [2012] and Azaria et al. [2013] provide an interesting analysis where the authors modify the recommender system to maximize the profit of the business. Our work is similar to the one presented in Azaria et al. [2013], but we reorder the YouTube recommender system to improve the efficiency of YouTube caches. This improvement in efficiency of caches may lead to profit of business for the content provider or distributor.

3. THE IMPACT OF VIDEO RECOMMENDATION

YouTube's related video list, shown in Figure 1, is a way to influence the decision of viewers on which video to watch next. When a viewer selects a video, a list of recommended videos is offered to the viewer on the same web page (in YouTube jargon defined as *watch page*), and the viewer may choose a video from that list. Our goal is to investigate if the related video list can be used to improve video distribution and delivery within YouTube. In order to understand the feasibility of using the related list to improve the performance of YouTube caches, we analyze a campus dataset containing YouTube requests to understand user behavior in watching YouTube videos.

3.1. Dataset Analysis

For the analysis of our proposed cache reordering approach, we analyze two network traces obtained from monitoring YouTube traffic entering and leaving the University of Connecticut campus network. The monitoring device is a PC with a Data Acquisition and Generation (DAG) card [NetworkMonitor 2014] which can capture Ethernet frames. The device is located at the campus network gateway, which allows it to see all traffic to and from the campus network. It was configured to capture a fixed length header of all HTTP packets going to and coming from the YouTube domain. The monitoring period for these traces was 72 hours each, and the general statistics are shown in Table I.

In trace T1, 47,986 video requests had the "related_video"[1] tag out of total 105,339 requests ($\sim$45%), which indicates that this video was selected by a viewer from the related video list. In the case of trace T2, the numbers are significantly lower since the trace was taken during winter break when the student population on campus is much lower, that is, $\sim$33% of the 7562 requested videos are selected from the related video list.

To investigate if users choose videos with a probability that decreases from the top to the bottom of the Related List, we further analyze data from these traces to determine the position of the related video selected by the users. We determine the position of a video that was selected by a viewer from the related video list as follows. We take the related video list for video A requested by a viewer and check if video B, requested by the same user, is in video A's related video list. If so, we determine the position of video B in video A's related video list. Figure 3 shows the results of this analysis for our

[1]YouTube previously had a related_video tag to identify if the videos were selected from the related list offered to the user. During the time of the collection of these traces, the tag was used. Now YouTube has removed the tag from their URL and parsing through the HTML page of original video is one way to know whether the next selected video is from the related video list.

Table I. Trace Characteristics

Trace file	T1	T2
Duration	3 days	3 days
Start date	Feb 6th 2012	Jan 8th 2010
# Requests	105339	7562
# Videos with "related_video" tag	47986 (45.6%)	2495 (33%)

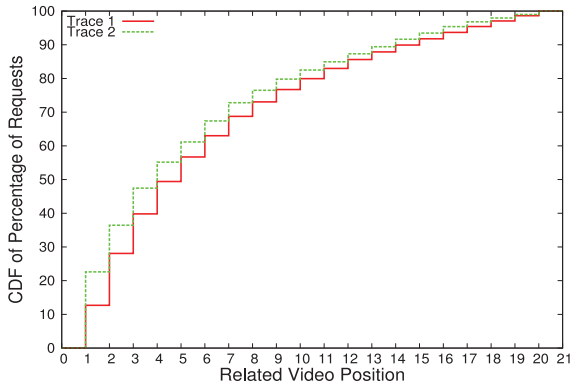


Fig. 3. CDF of the percentage of requests per related video postion for each video requested from the related video list.

trace (Trace T1), as well as the trace data used by Khemmarat et al. [2011] (Trace T2). Both traces show that about 50% of the viewers who make use of the related list, select an entry from the top five related videos. For the top ten videos, this number increases to 80%. This observation confirms our assumption about YouTube user behavior, that a user (potentially out of laziness) usually selects videos ranked higher on the related video list independent of the video content (but we assume also that the related list is originally sorted by relevance).

3.2. Chain and Loop Count

To better understand YouTube's recommendation system, we investigate how repeated selections of videos from the related lists behave. This has let us to define the concept of *loops* and *chains*.

The definition of *Loop Count* and *Chain Count* is as follows. *Loop Count* is defined as the number of consecutive videos a viewer requests from the recommended list before she requests the same video as the initial one and thereby forming a loop. *Chain Count* is the number of consecutive videos requested by the users from the recommended list until they request a new video by other means (e.g., a search).

Figure 4 provides an example of the definition of both Loop Count and Chain Count. In the example shown in Figure 4, both Loop Count and Chain Count is two.

The overall goal of this section is to gain better insight into the characteristics of how the usage of the related list influences the performance of caching.

3.2.1. Observed Chain Counts. In this section, we provide the results of the chain count analysis performed for the campus network traces described in Table I.

For the chain count analysis, we mine the requests from each user in the traces described in Section 3.1 to determine the number of consecutive requests made by the user from the related list of the previous video requested. The results from our chain count analysis on the two traces are summarized in Table II. For trace T1, the viewers

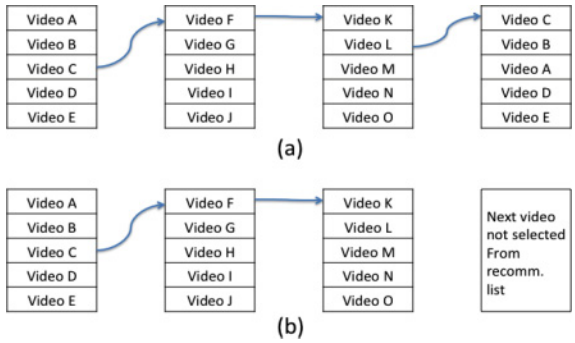


Fig. 4. Loop and Chain count example. (a) Example for a loop of length 2, (b) Example of a chain with length 2.

Table II. Chain Count Analysis Results

Approach	Trace T1	Trace T2
Avg. chain count	1.195	2.304
Max. chain count	8	21

choose videos from the related list at least once in 84.76% of the cases, and in 15.24% of the time, the chain count is larger than one, leading to an average chain count of 1.195. The maximum chain count seen in trace T1 is 8, that is, a user selects videos consecutively 8 times from the related list before selecting a video by other means. For trace T2, the values are a bit higher. In 48.20% of the cases, the chain count is at least one, and in 51.80% of the cases, it is more than one, with an average chain count of 2.304. The maximum chain count seen in trace T2 is 21.

The results in this section strengthens our assumption that the user selects videos from the related list and spends time on selecting videos from the related list consecutively. This user behavior of selecting from the related list consecutively works in favor of our related list reordering approach.

We could not find any loops in either of our traces. Therefore, to understand the YouTube recommendation system and generation of related list, we perform the loop count analysis on PlanetLab which is presented in the following section.

3.2.2. Loop Length at Fixed Positions. To investigate the loop count on a global scale, we performed a PlanetLab-based measurement. PlanetLab [2007] is a network of servers distributed around the world to perform experiments for the purpose of networking and distributed systems research. In a first experiment, 100 videos were accessed from each PlanetLab node. Then, we followed the chain of related videos by repeatedly selecting the same position. We did that for each of the positions 1 through 20. Making these deterministic selections, we continued to follow the chain until the loop closed and we arrived at the original video again. For example, if the top video is always selected from the related list and the initial video requested was selected again after four continuous selections from the related list, the loop count is three. The results of the experiment are shown in Figure 5.

The experiment was performed with possible regional deviations in mind, and the results shown in Figure 5 are subdivided by region. As can be seen from the figure, the general trend persists throughout the regions, and yields some rather surprising results.

It is understood that the related list is constructed from a search for related videos, and the result is then sorted by popularity before it is presented to the user. This

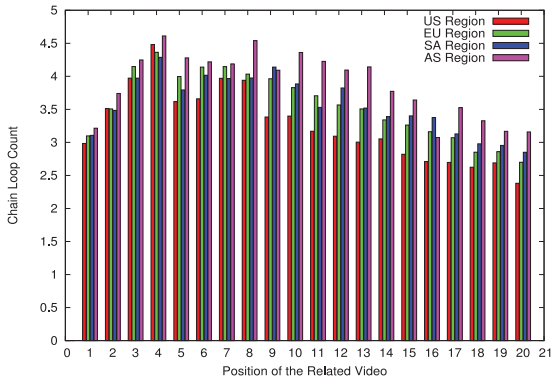


Fig. 5. PlanetLab-based loop count measurement for fixed position selection from related list.

understanding implies that videos that appear on lower positions (11–20) in the related list are less closely related to the current video than higher ones (1–10). Considering the large library of videos held by YouTube, we would expect that the set of less closely related videos is growing with distance; consequently, following the chain in the less closely (lower) related position should find a larger number of videos, leading to a longer loop. According to Figure 5, that is not the case.

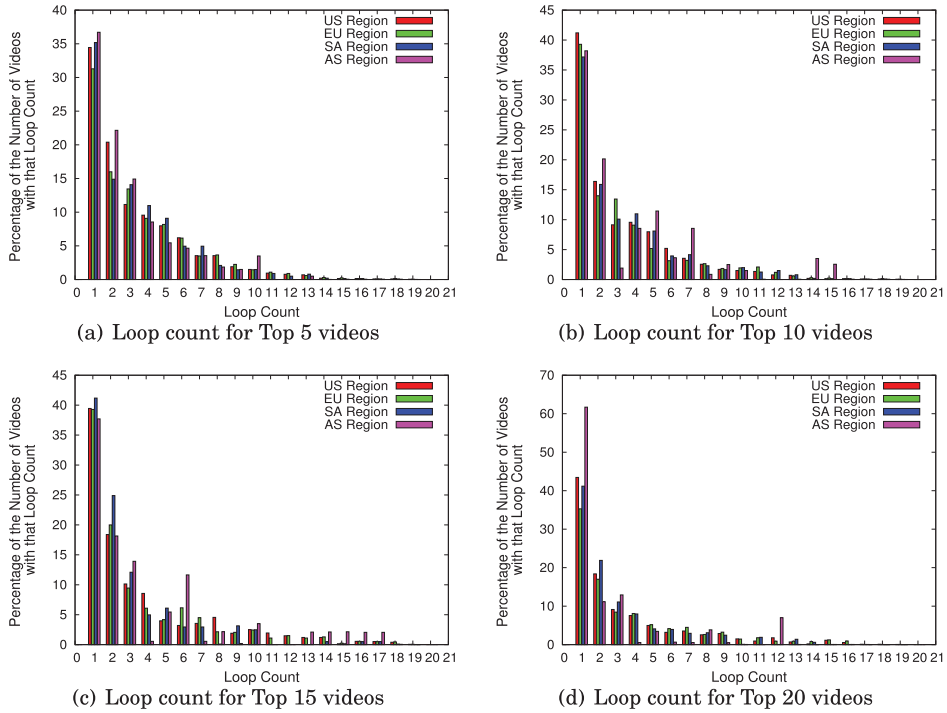
Since there could be many reasons for this behavior, we are trying to get a better understanding why the chain does not increase for lower related list positions. For that reason, we investigate the loop length for the case in which related list positions are randomly chosen.

3.2.3. Loop Length at Random Positions. Section 3.2.2 indicates that the pool of videos from which related lists are selected is rather small for each position. To better understand the impact of always choosing from a fixed position, we perform another experiment which is based on selecting videos from random positions on the related list. In this experiment, we vary the number of maximum positions that can be selected.

For this measurement, each PlanetLab node initially requests the same video and obtains the recommended list associated with this video. From this list, a new video is randomly selected and this procedure is repeated 50 times. This experiment is repeated four times where the selection process for videos from the recommended list changes for each experiment. In the first experiment, only the top 5 videos are randomly selected, in the second experiment the top 10, and in the third and fourth the top 15 and top 20 are selected, respectively. This procedure results in 50 video requests per node per experiment, and we use this result to determine the loop count.

The results from this measurement are shown in Figure 6. We decided to repeat the experiment with Top 5, 10, 15, and 20 video selection to investigate if different user behavior has an impact on the chain and loop length. The X-axis in Figure 6 shows the loop count and Y-axis shows the percentage of videos with the corresponding loop count in X-axis for Top 5 to Top 20 video selection. As can be seen in Figure 6, the trend for the loop lengths is relatively similar, independent of the range of top videos from which the selection is made.

It is noteworthy that the average loop length in case of random selection from a set of positions is similar to that of a fixed position, with minor changes for an increased number of maximum positions to choose from. The likely explanation for this effect is that the related list is mostly constructed from a limited and closed pool of videos. The search features that keep videos semantically close to each other dominate over features that would promote drift towards other videos. This is probably for the benefit



(a) Loop count for Top 5 videos

(b) Loop count for Top 10 videos

(c) Loop count for Top 15 videos

(d) Loop count for Top 20 videos

Fig. 6. PlanetLab-based loop count measurement for random selection from related list.

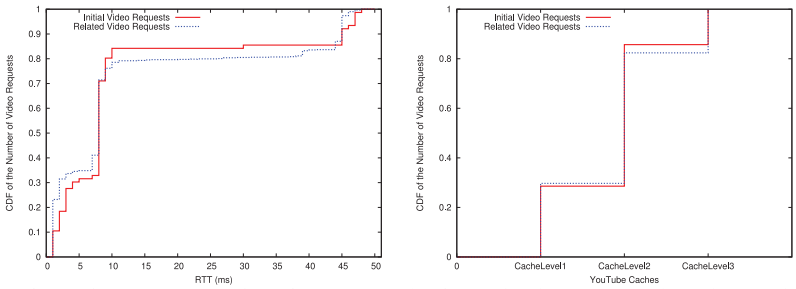
of the user: users selecting from the related list may not be interested in drift; the small pool prevents a "lost-in-hyperspace" effect. However, the limited pool size is also advantageous for the effectiveness of proxy caches. A request entering a pool through any video is apparently going to promote staying in the pool, increasing the chance that a chosen related video is in the cache.

Figures 5 and 6 indicate that YouTube does not assign the video to positions for some internal purpose. In particular, the mechanism that is used to compile the related list do not seem to be built to make caching more efficient. If that was the case, then the loop lengths shown in Figure 5 should be different from the one in Figure 6, which is not the case. Therefore, we can conclude that we can reorder the related list without interfering with the original goal of the YouTube related list.

3.3. Video Origin

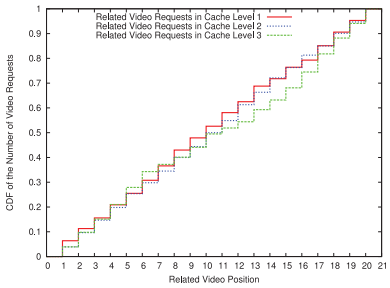
Since we are aware of the fact that YouTube is already employing caches in its video distribution infrastructure (see, e.g., YouTube [2012a] or Adhikari et al. [2011] for more details), we executed the following experiment to investigate if a video that is on the related list is served from a close by cache or not.

Our intention behind this idea is the conjecture that clearly distinct ranges of average RTTs for the TCP streaming sessions can be used to distinguish between different sources, and it is likely that larger RTTs indicate a larger distance (in hops) between source and client. To support this conjecture, we performed the following measurement from our campus network. First, we randomly selected 100 videos from trace T1 and retrieved each of the 100 videos while capturing a packet trace by using tcpdump [TcpDump 2010]. We then analyzed each of the 100 resulting packet traces and determined the average RTT of the TCP session that transmits the actual video data. We then



(a) CDF of average RTTs for TCP streaming session

(b) CDF of video requests based on cache levels



(c) CDF of video requests based on position of video in related list

Fig. 7. Distribution of video delivery based on response times and cache levels.

repeated this experiment for the top 20 videos from the related list of each of the initial 100 videos, resulting in 2000 data points.

Figure 7(a) shows the CDF of the average RTTs for each of the YouTube video requests as described earlier. As it can be seen from the figure, there are three different ranges of average RTTs (<3ms, between 3ms and 12ms and >12ms). Mapping these RTT ranges to IP addresses gives us three different subnet ranges. Thus, we can safely say that we see three different cache levels in our experiment and the CDF plot for the video requests from each of the cache levels (Cache Level1 is <3ms, Cache Level2 is between 3ms and 12ms and Cache Level3 is >12ms) is shown in Figure 7(b).

As shown in Figure 7(b), the majority for both groups of videos (initially selected and selected from the related list) are served from caches that are on level 2 (55% and 52%). Only ∼30% of the video from both groups are served from a level 1 cache. Further analysis revealed that all sessions with an average RTT of 3ms or less have a sender IP address from an address range that is administered by the University of Massachusetts, while address ranges for the level 2 and 3 caches are administered by Google (the owner of YouTube).

Based on the results presented before, we are now able to identify which of the videos selected from the related list are served from a close by cache (level 1) and which are served from caches further away (levels 2 and 3). With that information we determined the distribution of the position of the requested video in the related list for videos served from a close by cache and for those served from more distant sources. This distribution (in form of a CDF) is shown in Figure 7(c). The nature of the CDFs shown for both cases is almost uniform. This is a strong indicator that YouTube is not giving any preferences, in terms of caching based on the position of the video in the related list. It rather confirms the results presented in Adhikari et al. [2011, 2012] that claim

YouTube mainly focuses on load balancing between different cache levels and not on the goal of moving content as close to the client as possible.

Knowing that viewers are more likely to select top ranked videos on the related list, we are interested in researching if a caching approach that is tailored more towards this viewer behavior can improve caching efficiency. We present and investigate such an approach in Section 5.

4. RELATED LIST DIFFERENCES

In Section 3, the results from the chain count and loop count analysis of videos requested from the related list showed that the related videos are formed from a small pool of videos. In addition, we showed that YouTube does not assign related videos to nearby caches based on their position on the list to make caching more efficient. To further support our findings we look into the differences in related lists offered to clients by YouTube in this section. The goal is to understand if YouTube maintains the same related list across all clients in a region or clients across regions. The analysis results we present in this section also support the approach of caching and prefetching videos from the related list as proposed by Khemmarat et al. [2011]. The differences in related list also maintains our assumption that YouTube does not order the titles on the related list for any particular reason. Hence, reordering of related list based on the contents in the cache is not in any case an interference to YouTube's intentions of offering related list in a particular order.

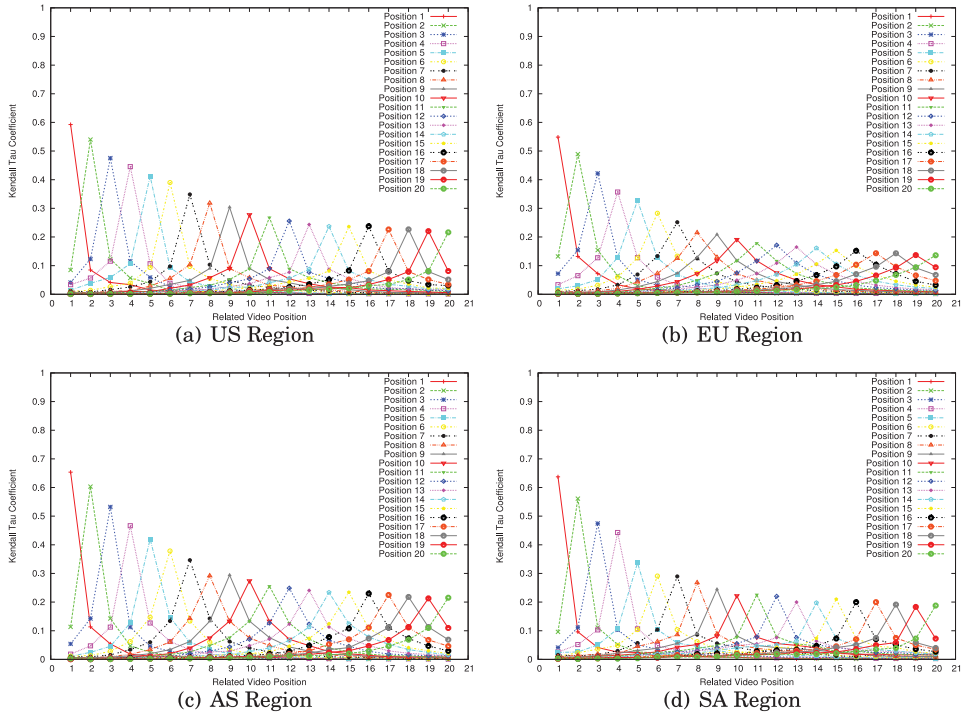
In the following, we present the experiment setup used in our analysis, the results obtained on the differences in the related list offered by YouTube and the impact of the related list differences on the effectiveness of caching videos from related lists. In this article, we only present the results and impact of regional related list differences for video requests from two different nodes in each region. Additional analysis results on related list differences are presented in Krishnappa et al. [2013a].

4.1. Experiment Methodology and Results

In this section, we describe the experiment setup and the measures we use to analyze the related list differences for a set of videos requested from PlanetLab nodes around the globe. Such related list changes can have a significant impact on caching and prefetching performance.

The goal of this measurement is to evaluate if the related list is always identical for all client requests for a specific video or if it changes (either between requests from different clients or between requests for the same video from the same client at different points in time). If the list changes frequently, this can lead to a significant amount of traffic between servers and caches. Thus, it is important to know how much of the related videos change from request to request, which allows us to identify how much of the related videos to cache or prefetch to improve the efficiency of caching and prefetching related videos. We decided to use PlanetLab nodes for our experiment, since it allows us to obtain global and regional information about related list changes.

To investigate differences between the related list for requests that originate from different nodes, we performed the following experiment. We first selected a set of PlanetLab nodes from four different regions (US nodes - 197, Europe (EU) nodes - 243, Asia (AS) nodes - 62, South America (SA) nodes -17, 519 in total). Each of these 519 PlanetLab nodes requests 100 YouTube videos randomly chosen from a trace collected in the US (see Section 3.1), and for each video request, we obtain the related list recommended by YouTube on its video *watch page*. To analyze the difference in related lists obtained from YouTube, we make use of the Kendall tau coefficient [KendallCoefficient 2014] as a measure of related list difference between two nodes in the same region.



(a) US Region (b) EU Region

(c) AS Region (d) SA Region

Fig. 8. Average content change-related video differences.

For each video, we perform a pairwise comparison for each related list position between related lists retrieved from all the PlanetLab nodes. For each related list position (1 to 20), if n PlanetLab nodes retrieve related lists of 100 YouTube videos, then there are $100 * n(n-1)/2$ comparison pairs. For each pair, we calculate the number of concordants if the pair of related list videos per position is similar or the number of discordants if they are dissimilar. The Kendall tau coefficient is calculated as $T = (Number of Concordants - Number of Discordants)/Total Number of Comparison Pairs$. In our measurement, we add 1 if the pair is similar (concordant) or 0 if they are dissimilar (discordant). Hence, T is always between 0 and 1 in our measurement instead of -1 to $+1$.

The Kendall tau coefficient analysis per position allows us to answer if there are more similar related list pairs at the top half of the related list or at the bottom half. The number of videos related to the current video must be expected to be rather big in many cases, while the list is always only a few videos long (usually 20). By performing a loop count analysis on the video requested by a user from related lists (Section 3.2), we have learned that not all potentially related videos are offered in the related lists by YouTube. With the Kendall tau coefficient, we can analyze whether the recommended list is derived from a single small pool of related video candidates, ordered by some mechanism, or whether videos higher on the recommended list are chosen from a smaller pool of more closely related videos.

Figure 8 shows the Kendall tau coefficient per related list position (1 to 20) for each PlanetLab region. As seen from each of the plots in Figure 8, the Kendall tau coefficient is high for the top half of the related list (positions 1 through 10) and remains constant for the bottom half (positions 11 through 20) of the related list. This result means that there are more dissimilar videos in the lower half of the related list provided for the

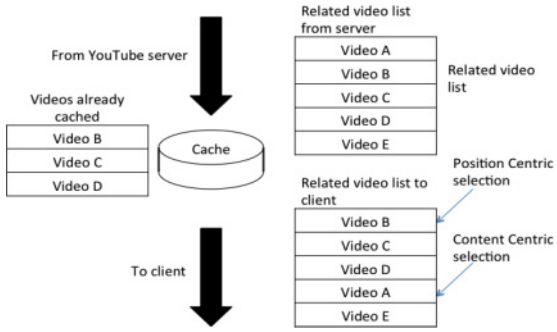


Fig. 9. Reordering of the related video list by the cache.

same video on different nodes of the same region. This result holds good for each region of the PlanetLab nodes selected.

The advantages of caching and prefetching of content on the related list were presented in detail by Khemmarat et al. [2011]. The given result showing that the Kendall tau coefficient is high for the top half of the related list shows that caching and prefetching of the top half of related list rather than bottom half reduces the amount of cache misses and does not incur unnecessary prefetching of related list videos.

In Krishnappa et al. [2013a], we present additional analysis results of related list differences by binning the related list and measuring the content change and order change of videos in the related list between nodes. Krishnappa et al. [2013a] also provides an in-depth analysis of related list differences for video requests from the same node on a daily basis and the impact of those related list differences.

5. LOCAL RECOMMENDATION TO IMPROVE CACHING

In this section, we present the general approach of related list reordering based on the contents in the cache, a study with 40 volunteer YouTube users to validate the hypothesis of users selecting from top of the reordered related list, and the results of our approach using real user traces.

From the results obtained in Section 3.1, we infer that the distribution of user requests on related video position is innate in the way users interact with the GUI. It should therefore be possible to use the order of the related video list for the purpose of increasing the cache hit rate.

The basic idea is to perform cache-based reordering, where the order of videos in the related video list is modified based on the cache's content. Figure 9 shows an example of the cache reordering approach. In this case, the related video list is sent from the server through the cache to the client. Upon receipt, the cache modifies the order of the related video list according to the videos it has currently cached. (Videos are neither removed from nor added to the list.) After modifying the list, the cache forwards the reordered list to the client.

The reordering maintains the order that the cached videos have in the original list as well as the order among the uncached videos. For example, as shown in Figure 9, this means, while B, C, and D are moved to the top of the related video list that is transmitted to the client, they (amongst themselves) keep the order they had in the related video list the cache originally received from the server.

Implementing the proposed cache-based reordering of the related video list is straight-forward and will add only a small amount of overhead on the cache. As shown in Figure 9, the reordering approach requires the cache to provide the following two functionalities: (a) maintain a list of the videos that are stored in the cache, and

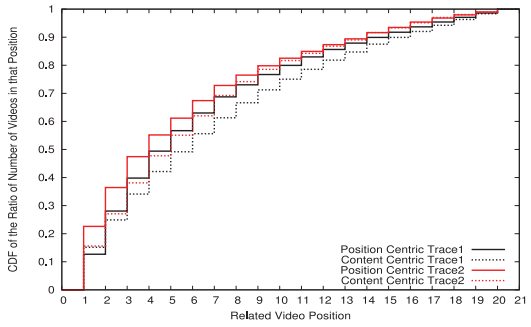


Fig. 10. Position of video selected from related video list after reordering.

(b) a process that is able to modify the related list. Functionality (a) will be provided by any ordinary cache since it needs to determine if a client request can be served locally or has to be forwarded to a server. The additional functionality (b) that is required for cache-based reordering is a simple manipulation of an HTML page that represents the related video list as, for instance, shown in Figure 1. Such simple modification of HTML code should increase the computational load on the cache insignificantly.

5.1. Related List Video Selection

In this Section, we discuss the two different choices of video selection from reordered related list and their impact on the reordering approach. The two choices are (i) content centric selection and (ii) position centric selection.

5.1.1. Content Centric. In the content centric approach, the requested video in the original trace before reordering the related list is fixed, that is, the sequence of videos that are requested is identical to the original trace. The position of the video that is requested from the modified related list, however, is different after reordering.

For the example shown in Figure 9, the viewer would select the video at position four of the related video list received from the cache, if the interest is absolutely in video A's content and not simply in one of the top listed videos. The content centric approach leads to a different probability distribution of the selected related video list position as shown in Figure 10. The dashed line in this figure shows the distribution of the positions that were chosen by a viewer in the case of content centric caching and a comparison with an unmodified related video list (see Section 3.1) as indicated by the solid line shows the difference in the two probability distributions. The comparison shows that the content centric approach leads to the selection of lower ranked videos. This is caused by the fact that noncached videos will be pushed down by the cache reordering scheme in the related video list forwarded to the client.

5.1.2. Position Centric. With the position centric approach, the related video list position selected by the viewer stays fixed. This clearly might result in a different content to what the viewer would have watched had the cache not modified the related video list (see Figure 9). In Figure 10, the solid line shows the distribution of position centric video selection from the reordered list for both the traces used in our analysis. The distribution is similar to the one in Figure 3, as we keep the position of the related video selected by the user same as in the original request.

Compared to the content centric approach, the position centric approach introduces two inaccuracies that we cannot account for without real-world testing. First, with more hits, the set of cached videos will be more stable, and more videos will be cached in competition with other content. This leads to an underestimation of reordering on

cache hits in our study. Second, reordering keeps popular videos at the top of the related video lists. The high probability of choosing videos from the top of the list leads the emulated user into cycles, which most real users would avoid. This leads to an overestimation of reordering on cache hits in our study.

To overcome these fallacies, we perform a user study to show that users select from the top of the related video list even after reordering and avoid entering cycles by searching different videos. We present this analysis in the next section.

5.2. User Study

Figure 3 showed that YouTube users select 80% of their next video from the top of the related list according to data collected from our campus trace. Our hypothesis is that users select from the top of the related list even after being provided with a modified or reordered related list, which suggests that we can modify the YouTube related list based on the contents in a cache to improve the cache efficiency of YouTube recommendation system. To verify our hypothesis, we perform a study with 40 actual YouTube users to investigate if a change of the related list is significantly changing the user behavior, in terms of most popular positions of which videos are selected from the related list. In the following, we describe the experiment we performed to gather data for this investigation and the results of the experiment.

5.2.1. Experiment Methodology. Our goal of this experiment is to collect empirical data that allows us to analyze how viewer behavior changes (in terms of selecting videos from the related list) if the original order of the related list is modified. We realize this experiment with the aid of a chrome browser plugin [YouTube 2014a], which intercepts the YouTube webpage at the client, randomly reorders the related list shown on the watch page and loads the modified page on the browser. The video currently requested by the client is not affected by the plugin. Apart from modifying the related list, the plugin also logs the video requested by the user and the modified related list offered to the user to our server on campus. Once the log is received we determine if the current video requested by the user belongs to the related list of the previous video requested from the same user. If so, then the position of the video requested on the modified related list is logged for our analysis. Since our goal is to verify if the user behavior of requesting videos from the top of the related list holds even after a modified related list is offered to them, the data collected from our experiment suffices.

In our experiment, no content is added or removed from the original related list sent by the YouTube server, only the order of the videos on that list is randomly changed. Since we modify the related list at the client using a plugin and not a central proxy which intercepts the whole traffic from YouTube, we decided to reorder the related list randomly (in contrast to the approach presented in Section 5). Also, the goal of the experiment is to prove our hypothesis that users request videos from the top of the related list even after the related list is reordered. Hence, building an actual proxy cache that would implement the mechanism for local recommendation is beyond the scope of this work.

5.2.2. Test Candidates and Results. To perform the experiment with a wide set of audience, we distributed the plugin among friends, colleagues, and students without informing them about the intentions of our investigation. We simply asked them to install the plugin in their browser and watch YouTube videos. We have 40 active users using the plugin and collected the logs for about 2 months with 11872 YouTube requests made by the end of January 2014. Out of the 11872 YouTube requests, 39.49% of requests are from the YouTube-related list. This confirms the results from our trace analysis presented in Table I.

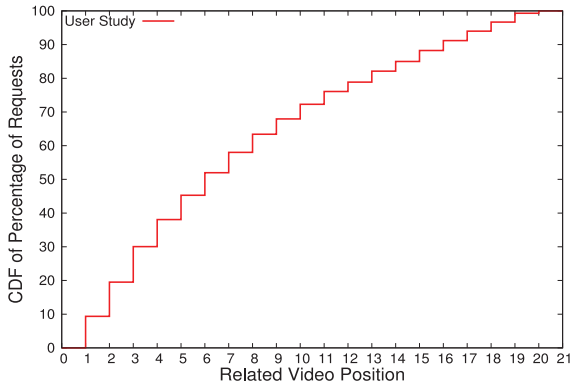


Fig. 11. CDF of the percentage of requests per related video postion for each video requested from reordered related video list.

As mentioned earlier, our objective of this user study is to verify our hypothesis that YouTube users do select videos from the top of the related list even after provided with a modified related list. Figure 11 shows the CDF of the percentage of requests per related list position for the 4688 requests made by YouTube users from the related list using our plugin. Figure 11 shows that 73% of the YouTube requests from the related list offered are from the Top 10 positions of the related list. From the traces captured from a university campus, the percentage of requests from related lists constitutes 80% of the requests, whereas this percentage is 73% in our user study that presents users with reordered related lists.

Visually, it is obvious from Figure 11 that the probability with which users select videos from the related list is similarly distributed as in the orignal trace. Consequently, we can say that even worst-case changes, that is, random permutation of the related list, does not alter user behaviour in a major way.

However, we cannot claim statistical identity. We performed the Durbin test [Conover 1999] on 4500 requests observed in the user study and on 4500 requests in trace T1, which was presented in Table I. The Durbin test rejects the hypothesis of identical distributions for original and permutated related lists, suggesting that lower entries in the related list are chosen more frequently after a random permutation reordering. Still, the dominance of the entries higher in the related list remains clear.

5.3. Results

We used the data from the campus network traces presented in Section 3.1 and information retrieved through the YouTube API YueTube [2007] to perform a simulation of the cache-based reordering. We simulate the caching of videos which were requested in the traces. For example, if we detect a request for Video A in the trace, we log this video as stored in the cache. Since each YouTube video is assigned a unique identifier which is included in the video requests in the network trace, we can accurately simulate the behavior of the cache. Via the YouTube API, we retrieve the related video list for each video that is requested in the network traces. Each time we simulate a video request we simulate the modification of the related list (retrieved via the API) based on what is stored in the cache. For example, if a video in the related list obtained from the YouTube API is present in the cache, the video is placed at the beginning of the re-ordered related list and the videos that are not present in the cache are pushed downwards. It is important to note that we do not change the content of the related list obtained from the YouTube API at all. Only the order of the videos in the related list is manipulated. We simulate user behavior where the viewer selects the video from the

Table III. Comparison of Hit Ratio

Trace	Content Centric	Position Centric
T1	6.71%	11.83%
T2	4.71%	22.90%

same position in the related list as presented in the original trace. In this scenario, the content the user selects at that position in the re-ordered related list might be different from the original content that was requested in the trace. For this investigation, we use a custom-built simulator implemented in perl. The results from our analysis are summarized in Table III.

Simulating a cache that employs the modification of the related video list with the content centric selection results in 7055 hits or a 6.71% hit ratio for trace T1 and 397 hits or a 4.71% hit ratio for trace T2. However, simulating a cache that employs the reordering of the related video list with the position centric selection results in 12431 hits or a 11.83% hit ratio for trace T1 and 1735 hits or a 22.90% hit ratio for T2. Although the hit ratio is not very high, this result along with our user study results from 5.2, affirming that users usually select from the top of the related list, shows the potential of our related list reordering approach to improve the performance of YouTube caches.

6. DISCUSSION

In this section, we discuss the advantages and disadvantages of cache centric video recommendation. We address the issues of increased cache complexity and server load reduction.

6.1. Why Only YouTube?

Our work is mostly concentrated on YouTube video delivery system and taking advantage of YouTube related list to improve the effectiveness of caches. While our work uses only YouTube traces to analyze the impact of our related list re-ordering approach, the idea remains valid for any video on demand system which offers short video clips and related lists to their users to choose their next video. There are a few VoD systems in the web which offers similar services to YouTube such as DailyMotion, Vimeo, Vine etc. However, none of them are as popular for hosting short video clips as YouTube. With over 1 Billion average monthly visits, 1 Billion registered users and presence in over 61 countries around the globe [YouTube 2014b], YouTube is by far the most popular and widely used user-centric video delivery system on the web. For these reasons, we have centered our approach and analysis of our re-ordering approach around YouTube. However, the idea is still valid for any video delivery system which offers short-clips and provides a related list for each video watched by their users.

6.2. Cost of Recommendation List Reordering

We have shown in Section 5.1.2 that the cache hit rate is significantly increased when the related video list is reordered according to the cache content (see Table III for detailed data on hit rates), using the access patterns from our campus network traces. The cost of this process is a more complex cache server. For each request, the server must identify URLs from YouTube and upon success, the cache must be inspected to find whether the requested video is cached or not. The cost of an additional cache lookup depends on the cache structure and size, but assuming a lookup structure based on a plain hash table, the average and worst case lookup times are $O(1 + n/k)$ and $O(n)$ respectively, where n is the number of elements in the cache and k is the number of buckets in the hash-table. Adding for example a second data structure for the bucket

list, like a self balanced tree, the theoretical worst-case time of common hash table operations can be brought down to $O(logm)$ rather than $O(m)$ where m is the number of elements in the hash-bucket.

Thus, the reordering comes at the described extra cache server cost. However, taking into account the very long tail distribution of videos, the gain in cache hit rate is substantial compared to the added processing. Additionally, there are several systems today that already rewrite web-page content. For example, Opera is currently rewriting and rearranging the content in web-pages using proxies for their opera mini browser [Opera 2012]. Similarly, proxies like FilterProxy [2001], Muffin [2012] and WebCleaner [2010] have the capability to modify content on the fly. We therefore claim that the proposed reordering of items in the related list is worth the additional complexity at the cache since this drawback is outweighed by the benefit of a significantly increased hit rate.

6.3. Reduction in Server Load

Another advantage of our related list reordering approach based on the content in the cache is the reduction in the server load and backend bandwidth consumption. With the reordering of the related list and pushing the contents from the bottom of the list to the top, we take advantage of the user behavior in selecting the videos from the top of the related video list. This also reduces the load on the YouTube server (or higher level caches) proportional to the number of cache hit gains. From the analysis in Section 5, we show that with our reordering approach, the cache hits increase using the T1 trace is from 6.71% to 11.83% which turns into an approximately doubling of the hit rate. Similarly, the bandwidth consumption reduces from 93.29% to 88.17%, which provides a 5.12% reduction in server load and backend bandwidth reduction from our related list reordering approach. Similarily, from the analysis of trace T2, we see a cache hit increase from 4.71% to 22.9% which is almost a five times increase on the hit rate. From the YouTube server point of view, this is a server load reduction of 18.19%.

6.4. Local Popularity Based Sorting of Related List Items

So far, the reordering of the related list does not take differences of local popularity of cached videos into account, since the reordering is based on a stable sort. This reordering could be further evolved by slightly modifying the sorting algorithm such that it also sorts the locally cached videos by their local popularity. Similar to the original reordering approach, the locally cached videos would still be pushed to the top of the related list. But the order in this group of videos would now be ordered by local popularity. For example, if video C is more popular than video B, then video C would be ranked first in the related video list that is sent to the client. This approach, however, needs further investigation to study its feasibility. First of all, the actual differences in popularity for videos that are stored on the local cache and belong to the related list of a video have to be determined. We believe that only significant differences in the popularity of the respective videos would render this approach feasible.

6.5. Implications of Re-Ordering

The re-ordering approaches presented in this article simulate a user selecting YouTube videos based on a related lists that have been modified based on the content of the cache that servers this user. While we investigate two different approaches (content centric and position centric as described in Section 5.1) and viewer behavior through a user study (Section 5.2), we cannot make an absolute claim that users will exactly follow either of the to approaches. Results from our user study indicate that the user behavior of requesting from the top of the related list does not change significantly even if the related list has been modified but a more in-depth analysis on the effect of re-ordering

on the video selection of users is required, which is quite challenging and outside the scope of this article.

7. CONCLUSION

Recent works have shown that, YouTube viewers select their next video request from the related list provided on the "watch page" of the current video with high probability. In this article, we took advantage of this user behavior to reorder the related list provided by YouTube based on the content already requested by the users in a gateway network. In our approach, the related list is modified only in the order of appearance on the user's webpage, not the content itself. In our reordering approach, the related videos present in the cache are moved to the top of the list and subsequently the related videos not cached are moved down. By analyzing a campus network trace filtered for YouTube traffic and a user study in which users were provided with modified related lists, we found that, users generally request videos from the top half of the related list, independent of the order of the related list. Hence, reordering the related videos already present in the cache makes our approach more advantageous.

We analyzed our approach in a simulation based study on the network trace captured on a campus gateway. We defined two different approaches of video selection from the re-ordered related list. Content centric selection, where the user selects the same video as he would have originally selected and position centric selection, where the user selects a video from the same position on the related list before reordering, which might change the original content. From our simulation study, we find that position centric selection of the reordered related list yields a better hit rate than content centric selection, which does not take advantage of the content in the cache. From the results obtained, we conclude that reordering the related list presented to the user based on the content already cached close to the users yields a better hit rate of the caches, thereby reducing the bandwidth consumption by multimedia requests and the latency in content delivery.

REFERENCES

V. K. Adhikari, S. Jain, Yingying Chen, and Zhi-Li Zhang. 2012. Vivisecting YouTube: An active measurement study. In *Proceedings of the 31st IEEE INFOCOM*. IEEE, 2521–2525. DOI:http://dx.doi.org/10.1109/INFCOM.2012.6195644

V. K. Adhikari, S. Jain, and Zhi-Li Zhang. 2011. Where do you "Tube"? Uncovering YouTube server selection strategy. In *Proceedings of the 20th IEEE ICCCN*. IEEE, 1–6. DOI:http://dx.doi.org/10.1109/ICCCN.2011.6006028

Amos Azaria, Avinatan Hassidim, Sarit Kraus, Adi Eshkol, Ofer Weintraub, and Irit Netanely. 2013. Movie recommender system for profit maximization. In *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM, 121–128.

Meeyoung Cha, Haewon Kwak, Pablo Rodriguez, Yongyeol Ahn, and Sue Moon. 2007. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *Proceedings of the Internet Measurement Conference (IMC'07)*. ACM, 1–14.

J. Chakareski. 2011. Browsing catalogue graphs: Content caching supercharged!!. In *Proceedings of the 8th International Conference on Image Processing*. IEEE, 2429–2432. DOI:http://dx.doi.org/10.1109/ICIP.2011.6116134

Xu Cheng and Jiangchuan Liu. 2009. NetTube: Exploring social networks for peer-to-peer short video sharing. In *Proceedings of the 28th IEEE INFOCOM*. IEEE, 1152–1160.

Xu Cheng, Jiangchuan Liu, and Haiyang Wang. 2009. Accelerating YouTube with Video Correlation. In *Proceedings of the 1st SIGMM Workshop on Social Media*. IEEE, 49–56. DOI:http://dx.doi.org/10.1145/1631144.1631156

W. J. Conover. 1999. *Practical Nonparametric Statistics* (3rd. Ed.). Wiley. 388–395.

James Davidson, Benjamin Liebald, Junning Liu, et al. 2010. The YouTube video recommendation system. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys'10)*. ACM, 293–296.

FilterProxy. 2001. HTTP Proxy. http://filterproxy.sourceforge.net/.

Google. 2012. Googletransparencyreport. http://www.google.com/transparencyreport/traffic/explorer/?r=US&l=YOUTUBE&csd=1389835560000&ced=1391045160000 (Last accessed Aug. 2014).

KendallCoefficient. 2014. Kendall Tau Rank Correlation Coefficient. http://en.wikipedia.org/wiki/Kendall_tau_rank_correlation_coefficient (Last accessed Aug. 2014).

Samamon Khemmarat, Renjie Zhou, Lixin Gao, and Michael Zink. 2011. Watching user generated videos with prefetching. In *Proceedings of the 2nd Annual ACM Conference on Multimedia Systems*. IEEE, 187–198.

Donald E. Knuth. 1998. *The Art of Computer Programming*, volume 3: Sorting and Searching (2nd Ed.). Addison Wesley Longman Publishing Co., Inc., Redwood City, CA.

Dilip Kumar Krishnappa, Michael Zink, and Carsten Griwodz. 2013a. What should you cCache?: A global analysis on YouTube related video caching. In *Proceedings of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 31–36.

Dilip Kumar Krishnappa, Michael Zink, Carsten Griwodz, and Pål Halvorsen. 2013b. Cache-centric video recommendation: An approach to improve the efficiency of YouTube caches. In *Proceedings of the 4th ACM Multimedia Systems Conference*. ACM, 261–270.

Muffin. 2012. World Wide Web Filtering System. http://muffin.doit.org/ (Last accessed Aug. 2014).

NetworkMonitor. 2014. Endace DAG network monitoring interface. http://www.emulex.com/products/network-visibility-products-and-services/endacedag-data-capture-cards/features/ (Last accessed Aug. 2014).

Opera. 2012. Mobile Browser. http://www.opera.com/mobile/ (Last accessed Aug. 2014).

Andreas Papadakis, Theodore Zahariadis, and George Mamais. 2013. Advanced content caching schemes and algorithms. *Adv. Electronics Telecommunications* 3, 5.

PlanetLab. 2007. PlanetLab Portal. http://planet-lab.org/.

Stefan Podlipnig and Laszlo Böszörmenyi. 2003. A survey of web cache replacement strategies. *ACM Comput. Surv.* 35, 4 (2003), 374–398.

YouTube. 2014a. YouTube Video Logger Chrome Plugin. https://chrome.google.com/webstore/detail/video-logger/nhilghfofbfdemgllaekjpkajmjemobb (Last accessed Aug. 2014).

YouTube. 2014b. Vimeo Vs YouTube. http://www.business2community.com/youtube/vimeo-vs-youtube-will-winner-emerge-2014-infographic-0864787 (Last accessed Dec. 2014).

TcpDump. 2010. Network packet analyzer. http://www.tcpdump.org/ (Last accessed Aug. 2014).

WebCleaner. 2010. A Filtering HTTP-Proxy.). http://webcleaner.sourceforge.net/ (Last accessed Aug. 2014).

Byungjoon Yoo and Kwansoo Kim. 2012. Does popularity decide rankings or do rankings decide popularity? An investigation of ranking mechanism design. Electron. Commerce Research Appl. 11, 2, 180–191.

YouTube. 2007. YouTubeAPI. https://developers.google.com/youtube/.

YouTube. 2012a. YouTube Keynote of MMSYS 2012. https://docs.google.com/presentation/pub?id=1bMLitOefxARBbgcu1v1xaJj89hbJGXYse17Xvgwro&start=false&loop=false&delayms=3000#slide=id.g47538e9_2_210 (Last accessed Aug. 2014).

YouTube. 2012b. YouTube press Statistics. http://www.youtube.com/yt/press/statistics.html (Last accessed Aug. 2014).

Renjie Zhou, Samamon Khemmarat, and Lixin Gao. 2010. The impact of YouTube recommendation system on video views. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 404–410. DOI:http://dx.doi.org/10.1145/1879141.1879193

Renjie Zhou, Samamon Khemmarat, Lixin Gao, and Huiqiang Wang. 2011. Boosting video popularity through recommendation systems. In *Proceedings of Databases and Social Networks*. ACM, 13–18. DOI:http://dx.doi.org/10.1145/1996413.1996416

Michael Zink, Kyoungwon Suh, Yu, and James Kurose. 2009. Characteristics of YouTube network traffic at a campus network: Measurements, models, and implications. *Elsevier Computer Networks* 53, 4, 501–514.

Received March 2014; revised August 2014; accepted December 2014