Journal of Network and Computer Applications 69 (2016) 40–53

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Multimedia prefetching with optimal Markovian policies

Cezar Pleşca [a,*], Vincent Charvillat [b], Wei Tsang Ooi [c]

[a] Department of Computer Science, Military Technical Academy, 81–83 Bd. George Cosbuc, Bucharest, Romania
[b] Department of Computer Science and Applied Mathematics, IRIT-ENSEEIHT, 2 rue Camichel, 31071 Toulouse, France
[c] Department of Computer Science, School of Computing, National University of Singapore, Singapore 117417, Singapore

ARTICLE INFO

Article history:
Received 18 May 2015
Received in revised form
20 March 2016
Accepted 4 May 2016
Available online 7 May 2016

Keywords:
Multimedia
Prefetching
Markov Decision Process
Profiling
Uncertainty
Optimization

ABSTRACT

Multimedia prefetching is able to reduce the end-to-end latency and improve the video quality perceived by users. Previous work modeled prefetching as applying sequential decisional policies under uncertainty, using a Markov Decision Process model that integrated both user behavior and resources availability, to achieve optimal prefetching policies under realistic assumptions. In this paper, we extended the existing MDP model by considering more complex and aggressive policies, while preserving the optimality of the prefetching policy. We further enriched the extended model by considering user's profile to provide finer prefetching policies. The proposed extensions and the associated policies are validated through comparison against the existing model and against heuristics found in related work. We showed that our profiles-aware optimal policies can be achieved up to 28% latency reduction with respect to known heuristics.

1. Introduction

Cisco (2015)predicted that 80% of all consumer Internet traffic (slightly more than one zettabyte) will be video traffic by 2019, about 4 times the video traffic in 2014. This trend is driven by popular services and Web sites, such as Netflix and YouTube, offering video contents ranging from home videos, news clips, and product advertisement, to professionally produced TV and movies. Users of these websites typically are presented with a set of video clips to view. The user selects one that she or he would like to view, causing the video to be streamed to the web browser. Once a sufficient amount of video data has been buffered, the video playback starts, while the rest of the video clip is still being downloaded. After watching a video, the user is typically presented with a selection of video clips that the system thinks the user may like (i.e., related list or video recommendations). The user might select one video from this set of video clips to watch, and the process repeats.

Recently, Krishnappa et al. (2013b) showed that viewers frequently choose videos from the related list and that the order of recommended video clips does influence the users' navigation patterns. In their work, these authors used this property and proposed to reorder recommended videos (by YouTube), prioritizing the ones that are already cached and ready to be delivered to the users. This strategy improved the viewing experience while jumping from one video clip to the subsequent one with reduced (resp. increased) latencies (resp. cache hit rates). Moreover, for a group of related videos, it is likely that a viewer would watch next video from the related list after viewing the previous one. Therefore, once a video is played back by a user, subsequent videos could be prefetched. Combining caching and prefetching is useful for popular applications such as online TV and video streaming services (Krishnappa et al., 2011a). As an example, Liang and colleagues recently introduced an integrated prefetching and caching proxy for services like Netflix and YouTube (Liang et al., 2015). These authors designed prefetching strategies that reduce latency between content servers and proxies in HTTP-based adaptive video streaming. In general, caching and prefetching are the two principal system techniques used to reduce latency. Both techniques have been extensively studied to reduce web pages access latency. The techniques complement each other: caching keeps a copy of downloaded objects in anticipation of repeated access, while prefetching downloads objects using any spare bandwidth in anticipation of future access.

In this work, we aimed at theoretically characterizing optimal prefetching policies. We considered users' navigation patterns within a video collection. While a user jumps from one video to another one, we tried to minimize the cumulative latency due to buffering of each video within an access sequence.

Accurately predicting which video will be accessed in the near future is a key to achieving good prefetching performance. A wrong prediction would lead to downloading videos chunks that

* Corresponding author.

http://dx.doi.org/10.1016/j.jnca.2016.05.002
1084-8045/

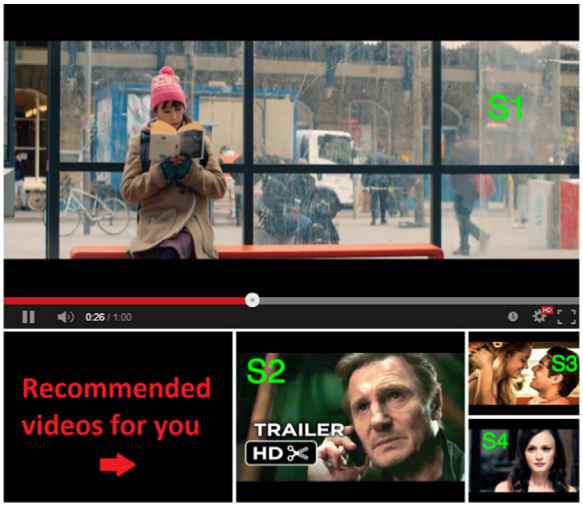

Fig. 1. Video content with related clips in the bottom-side of the video panel.

are not needed and does not reduce latency. Fortunately, predictions are often statistically possible due to the reproduction of users' navigation patterns. Frequent patterns might occur due to popular contents within specific communities of users. Frequent patterns might also be a direct consequence of an interface bias, such as the play lists on YouTube, for example. Fig. 1 illustrates another type of interface bias: even if the three related clips at the bottom are recommended, the left one is usually chosen more frequently because of its position and size.

Existing techniques for prefetching are often heuristically designed due to the complexity of the problem as discussed by Morad and Jean-Marie (2014a). Beyond the access pattern prediction, both the available resources (e.g., spare network bandwidth) and video clips sizes matter. Variability among users (e.g., profiles from different geographic regions, Krishnappa et al., 2013a) should also be understood by a smart prefetching agent. When several user profiles coexist, a simple prefetching policy can only fit with one mixed (e.g., averaged) profile, leading to sub-optimal results. We tackled this issue in this work. The core of this paper significantly extended our initial model by Charvillat and Grigoras (2007), already published in this journal. Our contributions in this paper are the following:

1. We proposed a general model that encompasses many existing techniques for prefetching. Our model allows computation of optimal Markovian prefetching policies based on complex access patterns and any network condition.
2. Our model inherently/automatically classifies the user access patterns into different profiles, allowing more accurate predictions and thus better prefetching performance.

We structured the rest of the paper into seven sections. Section 2 describes a general framework for prefetching, and presents several prefetching policies. Section 3 recapitulates our initial model for deciding the optimal prefetching policy using an approach based on Markov Decision Process. Section 4 extends our model by considering multiple stream prefetching policies while Section 5 considers multiple user profiles. We evaluate our proposed prefetching method in Section 6. Section 7 contrasts our work with existing literature about prefetching. Finally, Section 8 concludes the paper by summarizing the key messages of this paper and reflecting on possible future directions of this research.

2. Prefetching model

In this section, we describe, informally, our general model for prefetching video clips. In our model, the user is presented with a set of recommended video clips to watch, either while watching a video (see the screen shot in Fig. 1), or after watching a video. Prefetching decision is made after a prefetching operation completes, or when the user selects a new video to watch.

For each clip prefetched, we always prefetch only the initial segment of the video, corresponding to the buffering time required to playback the video. In other words, a video whose initial segment has been prefetched completely can start playing immediately if the user selects this video to watch later. For brevity, when we say that a *video has been prefetched*, we mean that the *initial segment* of the video has been prefetched.

It may happen that the initial segment of a video is only partially prefetched. This situation occurs when the user selects a new video to watch, while the other videos are still being prefetched. In this case, a latency is incurred if the user decides to watch the partially prefetched video later. The user would have to wait for the initial segment of the video to finish buffering before playback starts.

A good prefetching policy should reduce the expected latency experienced by users while they watch the video clips from a given video collection. We use the expected latency as the performance objective. Another commonly used objective, the hit ratio (how many objects prefetched has been accessed) is not really suitable, as it does not consider partially prefetched objects.

A prefetching action should also consider how much of the spare bandwidth to allocate to prefetch each video. The prefetching decision depends on several factors. The first factor is the user behavior. The prefetching agent should prefetch videos that are likely to be accessed by the users after watching the current video. For instance, on YouTube, a long video is commonly split into shorter video clips. In this case, users are likely to watch the video in their original sequence. The second factor is the amount of video that we can prefetch within a given time. This factor in turn depends on the total spare bandwidth, the download rate, and the playback rate of the video. The third factor is how much of the initial segment of a video has been partially prefetched before.

The situation illustrated by Fig. 1 naturally leads to several intuitive prefetching heuristics. If the video stream S_2 is most likely to be viewed by the user after S_1, the associated transition probability $p(S_2|S_1)$ should be higher than $p(S_3|S_1)$ or $p(S_4|S_1)$. An intuitive idea is then to prefetch either S_2 only, or S_2, S_3 and S_4 proportionally according to their transition probabilities. Beyond this, several other policies can be devised. To allow the explanation of these heuristic policies, we first model the user behavior as a weighted directed graph called the *navigation graph* (or execution digraph as named by Fomin et al., 2014), where each vertex is a video and an edge going from video S_i to video S_j with weight $p(S_j|S_i)$. Fig. 2 shows an example of the navigation graph.

Let S_{cur} be the video currently being watched by the user. Every time the user selects a video S_{cur} to watch, the entire available bandwidth is allocated to download the initial segment of S_{cur}, to reduce the buffering time. Any on-going prefetching will stop. When the initial segment of S_{cur} is downloaded and the playback of S_{cur} starts, S_{cur} will be downloaded with data rate equivalent to the video playback rate. If there are spare bandwidth on the network, then prefetching of other video streams starts in the background, according to some policy. We will further introduce some well known heuristics policies.