On the Feasibility of Prefetching and Caching for Online TV Services: A Measurement Study on Hulu

Dilip Kumar Krishnappa, Samamon Khemmarat, Lixin Gao, and Michael Zink

University of Massachusetts Amherst, USA
{krishnappa,khemmarat,lgao,zink}@ecs.umass.edu

Abstract. Lately researchers are looking at ways to reduce the delay on video playback through mechanisms like prefetching and caching for Video-on-Demand (VoD) services. The usage of prefetching and caching also has the potential to reduce the amount of network bandwidth usage, as most popular requests are served from a local cache rather than the server containing the original content. In this paper, we investigate the advantages of having such a prefetching and caching scheme for a free hosting service of professionally created video (movies and TV shows) named "hulu". We look into the advantages of using a prefetching scheme where the most popular videos of the week, as provided by the hulu website, are prefetched and compare this approach with a conventional LRU caching scheme with limited storage space and a combined scheme of prefetching and caching. Results from our measurement and analysis shows that employing a basic caching scheme at the proxy yields a hit ratio of up to 77.69%, but requires storage of about 236GB. Further analysis shows that a prefetching scheme where the top-100 popular videos of the week are downloaded to the proxy yields a hit ratio of 44% with a storage requirement of 10GB. A LRU caching scheme with a storage limitation of 20GB can achieve a hit ratio of 55% but downloads 4713 videos to achieve such high hit ratio compared to 100 videos in prefetching scheme, whereas a scheme with both prefetching and caching with the same storage yields a hit ratio of 59% with download requirement of 4439 videos. We find that employing a scheme of prefetching along with caching with trade-off on the storage will yield a better hit ratio and bandwidth saving than individual caching or prefetching schemes.

Keywords: Video-on-Demand services, Hulu, Cache, and Prefetching.

1 Introduction

The Internet has emerged as a prime medium for TV shows, radio programs, movies, and the exchange of videos for personal as well as commercial use. The advent of websites such as Hulu [1] and Netflix [2], which offer streaming of TV shows and movies, has made the Internet a major source for digital entertainment in the US. The growing use and popularity of content streaming among users is closely tied to the increasing popularity of broadband Internet connection in homes. The greater adoption of broadband in the US has motivated television channels such as NBC and ABC to offer their prime-time programming to online viewers via the media content provider hulu. In parallel, Netflix, a DVD rental company began to take advantage of the click-and-view streaming of full-length films and television episodes with a subscription service.

In the measurement study described in this paper, we focus on hulu as it is free and offers ad-supported streaming video of TV shows and movies from NBC, Fox, ABC, and many other networks and studios [3]. The advantage of hulu is that it is owned by these corporations, and the shows that air on their traditional TV channels are available for Internet users the next day for free (but not free of ads). This is popular in university campuses as many students would not have a TV in their dorm rooms and rely on Internet content for entertainment. Apart from TV shows, movies and video clips from other commercial sources are also hosted for free on hulu.

Due to the high popularity of TV shows and movies hosted on hulu, many people watch the same content in a certain time period. Our analysis of how hulu requests are distributed reveals that the requested videos are streamed from original servers hosting the content even when multiple clients request the same video, which shows that there is no proxy employed. This redundancy in streaming the same video from a server which is far away leads to an unnecessary increase in the network traffic.

In this paper, we investigate, through trace-based simulations, how prefetching and caching of videos requested from a campus network could reduce the consumption of network bandwidth by reducing multiple downloads of the same video from the origin server(s). We evaluate three different schemes: conventional caching scheme, popularity based prefetching scheme [5] and a combined scheme. The popular videos list is obtained from the hulu website, which is updated on a weekly basis. In our popularity-based prefetching simulation, we download the top-100 videos from that list to our local cache. Next to reducing bandwidth consumption, prefetching and caching can also reduce the potential of delayed playout, and pauses during video playback since videos streamed from the proxy are not prone to congestion or outages in the backbone network.

We evaluate the proposed caching and prefetching schemes with user browsing pattern data collected from a university network. Results from our trace-driven simulation show that a conventional caching scheme at the proxy with no limit on storage yields a hit ratio of up to 77.69%. A prefetching scheme where the top-100 popular videos of the week are downloaded to the proxy yields a hit ratio of 44% with a storage requirement of 10GB and download requirement of 100 videos. A LRU caching scheme with a storage limitation of 20GB can achieve a maximum hit ratio 55% % but downloads 4713 videos to achieve such high hit ratio compared to 100 videos in prefetching scheme, whereas a scheme with both prefetching and caching with the same storage yields a hit ratio of 59% with download requirement of 4439 videos. We find that employing a prefetching scheme along with caching with limited storage will yield a better hit ratio than individual caching or prefetching schemes.

Although caching and prefetching are not new mechanisms [6, 7], we believe that, to the best of our knowledge, our work is the first that systematically investigates their effectiveness on the hulu VoD service based on trace-driven simulations.

2 Methodology

In this section, we describe our methodology to monitor the traffic between clients in our campus network and hulu servers. The methodology allows us to understand how a client receives a video stream from hulu and to obtain the hulu usage statistics in our campus network. Also, we explain the extraction of hulu requests from the captured trace.

The measurement equipment used to monitor the traffic between clients in our campus network and hulu servers is a commodity PC installed with a DAG card [4] to capture packet headers. It is placed at the gateway router of UMass Amherst, connected via optical splitters to the Giga-bit access link connecting the campus network to a commercial ISP. The TCP and IP headers of all the packets that traverse these links are captured by the DAG card along with the current timestamp. In addition, we capture the HTTP headers of all the HTTP packets going out to www.hulu.com. Note that all the recorded IP addresses are anonymized. (A more detailed description of the measurement setup can be found in [8].)

For each outgoing packet through the gateway router, its timestamp, source IP address, destination IP address and the HTTP request header are extracted from the captured trace files. Out of these packets, the ones containing only hulu requests are filtered using the filtering pattern "/watch/" and the destination IP address of hulu servers. The video requests that are unique in the trace were filtered using sort and eliminate duplicates algorithm to obtain information about the number of duplicate requests present in the trace.

3 Dataset

In this section, we present the dataset obtained by the measurement process described in the previous section.

Table 1. Day-to-Day statistics of the trace

Trace	Total Video Requests	Unique Videos	Percentage (%)
Day1	3511	1109	31.58
Day2	3461	1101	31.81
Day3	3616	1113	30.77
Total	10588	2363	22.31

3.1 Trace Details

For our analysis we captured a three day network trace using the measurement setup described in Section 2. The trace was captured during fall 2010 semester when students were back in full numbers. The trace captured was filtered for hulu data as explained in Section 2. There were 10,588 hulu video requests in a three day period where only 2,363 distinct videos were requested in total. Table 1 provides the day-to-day and total statistics of the hulu trace used in our analysis. It should be noted that the total unique videos value of 2,363 is not the sum of the unique videos of each day as seen from the table. This is an artifact of subdividing the trace into single day data and shows that videos are repeatedly requested not only in a 24-hour time span but also over several days. The table also shows that there are only 22.31% distinct video requests, which leaves us with 77.69% of the video requests being two or more requests for the same video. This is an important result since this indicates the feasibility of prefetching and caching.

To give an overview of the usage of hulu on campus, we use the trace details to show the number of requests for each unique video during the period of the trace. Figure 1 shows the CCDF plot of the popularity graph describing the requests per video similar to [8]. We can see that the number of unique videos requested only once are about 48.92% (1,156 videos), which leaves us with a majority 51.08% (1,207 videos) requested multiple times, demonstrating the popularity of the content provided in hulu.

3.2 Popular Video List Details

In addition to the network trace, to validate our proposed prefetching approach, we obtain the list of most popular videos watched by viewers for a particular week preceding the capture of the traces. The hulu website provides a list of videos which are ranked in the order of their popularity for a particular day, week or month. We chose the weekly popularity list since many TV shows are updated on a weekly basis rather than daily or monthly basis. Our experiment shows that change in popularity of videos over a week is minimal. Thus, popularity list on a weekly basis serves best for prefetching. We use 'wget' to obtain the HTML page that contains popular videos list from the hulu website. We then parse the obtained HTML page to extract the URLs of the popular videos. These data are later used to simulate the prefetching of the videos from the hulu server to our local storage.

4 Simulation and Results

In this section, we present a simulation methodology for the evaluation of our proposed approaches. Through trace-driven simulations, we compare the performance of the cache-only and prefetch-only schemes. We also evaluate the performance of an approach that combines both caching and prefetching. Also, the impact of storage size on the performance of our proposed schemes and the overall bandwidth consumption is evaluated.

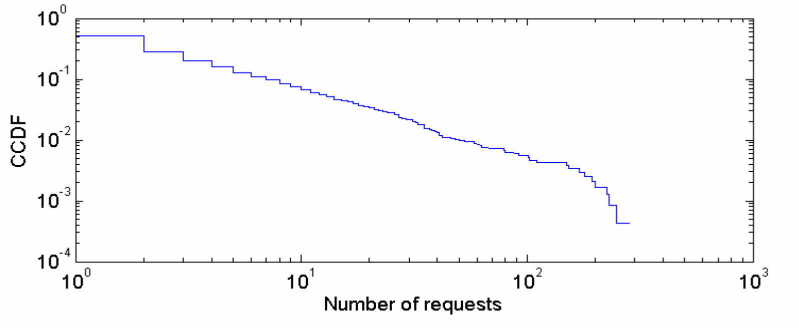


Fig. 1. CCDF popularity plot of the hulu trace

4.1 Evaluation Metrics

We simulate the proposed prefetching and caching schemes from real user request patterns by issuing video requests based on the network trace presented in Section 3.1. Prefetching is simulated by maintaining a prefetching storage which keeps track of the list of popular videos list obtained from the hulu website. Similarly, the caching scheme is simulated by providing storage on the proxy which holds the videos requested by viewers, if not already present in the storage.

We perform our simulation of the caching scheme for cases where the storage space is unlimited and also the case where there is limited storage space. For simplicity, the storage space size is defined by the number of slots where each slot can hold one hulu video. Based on our measurement on the size occupied by HD hulu video, it is approximated as each hulu video requires about 100MB of space, which corresponds to the size of each slot in our storage.

In this study, we use hit ratio as the metric to evaluate the proposed prefetching and caching schemes. Hit ratio is defined as a fraction of the number of requests for a video that can be served from the prefetching/caching storage (called hit requests) over the total number video requests.

$$\textit{hit ratio} = \textit{hit requests/all requests}$$

A higher hit ratio means we can serve more requests from the prefetching/caching storage, resulting in a reduction of bandwidth usage.

4.2 Performance of Caching without Storage Limit

We first present the performance of the caching scheme without any limit on the storage required to cache the videos. The caching scheme is simulated as follows: Each video requested by the user is downloaded to the local proxy placed on the edge of the campus network[1]. Video requests from clients are directed to the proxy. If the video is already cached at the proxy, it will be streamed from here; if not, the request is forwarded to the hosting server, and the video is streamed from the server through the proxy to the requesting client. Using this scheme a hit ratio of 77.69% is obtained. Although this scheme provides a very high hit ratio, the amount of storage required increases significantly as the number of video requests from clients increase. To implement this scheme, 236GB storage would be required, which corresponds to the 2,363 unique videos present in our trace. Also, the amount of bandwidth required to download all the videos into the local storage increases with the number of unique videos. Though 236GB storage seems reasonable, when this approach is applied to a bigger access network or a week-long trace, the amount of storage required increases considerably. Thus, this scheme is not necessarily feasible for implementation on a larger network.

[1] For all caching schemes mentioned in this paper we assume so called "write-through" caching [9]. In this case, a video that's not already cached is streamed from the origin server through the proxy to the requesting client.

4.3 Performance of Caching With Storage Limit

Next, we present the evaluation results for a caching scheme that is slightly modified from the one presented in Section 4.2. In comparison to the previous approach, storage on the proxy is now limited. Let N represent the number of videos that can be cached in the storage. We evaluate this scheme by varying N from 100 to 2000 which corresponds to varying the storage limit from 10GB to 200GB. Figure 2(a) shows the resulting hit ratio of such a scheme. Once the storage limit is reached, LRU caching scheme is employed to remove the least accessed video.

The figure shows that the hit ratio increases gradually for small storage spaces till N=1000 after which the increase in hit ratio is minimal as we increase the number of videos that are cached and reaches the maximum hit ratio of 77.69% as in case of caching without storage limit. As seen from Figure 2(a), a storage limit of 50GB will yield a hit ratio of 67%, while doubling the storage space yields a hit ratio of 73.86%. Though the improvement in hit ratio is minimal, the amount of bandwidth savings is increased as we increase the storage space.

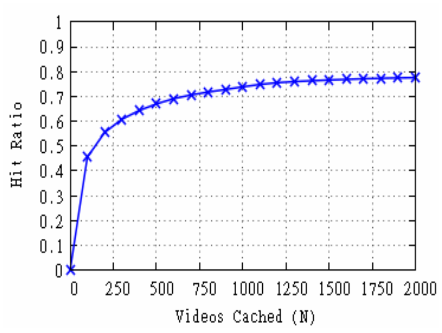
For example, the number of videos that need to be streamed[2] from the origin server to obtain a hit ratio of 67% which corresponds to the storage size of 50GB is 3494, whereas this number decreases to 2767 (resulting in a hit ratio of 73.86%) when the storage size on the proxy increases to 100GB. Thus, increase in storage space yields higher hit ratio and bandwidth savings. Also, there exists a trade-off between the hit ratio desired and storage space provided.

4.4 Performance of Prefetching Popular Videos List

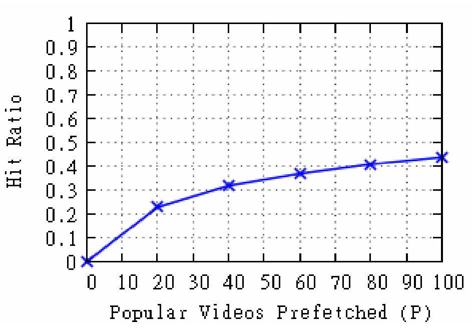
After analyzing the limited and unlimited caching scheme, we now evaluate the performance of prefetching the popular videos list obtained as explained in Section 3.2. Let P represent the number of popular videos prefetched. We evaluate this scheme by varying P from 20 to 100 which corresponds to varying the prefetching storage from 2GB to 10GB. Figure 2(b) shows the hit ratio of such a scheme.

The figure depicts the variation of hit ratio with the increase in prefetching of most popular videos of the week from 20 to 100. It can be observed from the figure that the hit ratio increases gradually till P = 60, and then the increase in hit ratio is relatively minimal. The maximum hit ratio of 44.2% is obtained when P=100 which corresponds to storage space of 10GB. Though the LRU caching scheme as mentioned in section 4.3 yields a hit ratio of 45.53% for the same storage space, the important point to be noted in this evaluation is the fact that the number of videos downloaded to the prefetch cache is just 100 compared to 5767 videos in case of LRU cache. Thus the amount of bandwidth savings is very high in prefetching scheme compared to the caching scheme.

[2] The amount of videos downloaded is not proportional to the numbers mentioned in Table 1. Videos are downloaded only when LRU scheme decides to remove a video due to space constraint.



(a) Caching Scheme



(b) Prefetching Scheme

Fig. 2. Hit Ratio with varying storage limits

In addition, our simulation shows that 100% of the popular videos from P = 20 to P = 60 list were requested by the clients, whereas it is 95% for P = 80 and P = 100. This shows that almost all videos in the top-100 popular videos list are watched at least once by the clients in a three day period of our trace. Also the change in the popular videos list is minimal over a week period as we consider the popular videos of a week in our analysis. Thus, it is feasible and advantageous to implement the prefetching of popular videos scheme.

4.5 Combining Caching and Prefetching

In the previous section, we have shown that the bandwidth savings that can be obtained with the prefetching scheme is high. On the other hand, the videos served by the top-100 videos prefetched at the proxy are only 44.2% of the total requests, which leaves us with more than half of the videos in the trace left unattended by the prefetching scheme. Some of these unattended videos from the prefetching scheme can be taken care of by employing a caching scheme. Thus, the combinination of prefetching and caching schemes called prefetch-and-cache scheme serves more videos and uses less bandwidth than individual schemes.

The simulation of the combination of caching and prefetching scheme is carried out as follows: (i) a storage is maintained on the proxy with a fixed part and a variable cache part. The fixed part of the storage holds the prefetched popular videos. (ii) all user requests are directed to the proxy. The video requested is searched for both in the prefetch or cache part of the storage (iii) if the video requested by the user is not present in the storage, then the request is sent to the hulu server hosting the video. The resulting stream from the hulu server is cached in the variable part of the storage. (iv) if the variable part of the storage is filled, videos are removed from the variable part of the storage using LRU scheme.

Figure 3 shows the hit ratio resulting from the prefetch-and-cache scheme. The combination of two schemes increases the hit ratio by 3-5% for the same amount of storage as in the caching-only scheme. For example, a storage limit of 20GB in caching-only scheme will hold about 200 videos and yields a hit ratio of 55.5% as seen in Figure 2(a). The same storage limitation in prefetch-and-cache scheme with 100 videos prefetched and 100 videos cached would yield a hit ratio of 59%, which is a slight improvement over the caching only scheme.

The combination is also an improvement over the prefetch-only scheme. As seen, the prefetch scheme offers a maximum hit ratio of 44.2% and the other videos cannot be served by employing prefetching scheme. By combining both prefetching and caching, all the requests by the clients can be served from the cache with increase in hit ratio compared to prefetching only or caching only scheme. Again it is a trade-off between the storage available and the hit ratio desired, but the advantage of this combination scheme is that the storage required to obtain the desired hit ratio is less than the cache-only scheme.

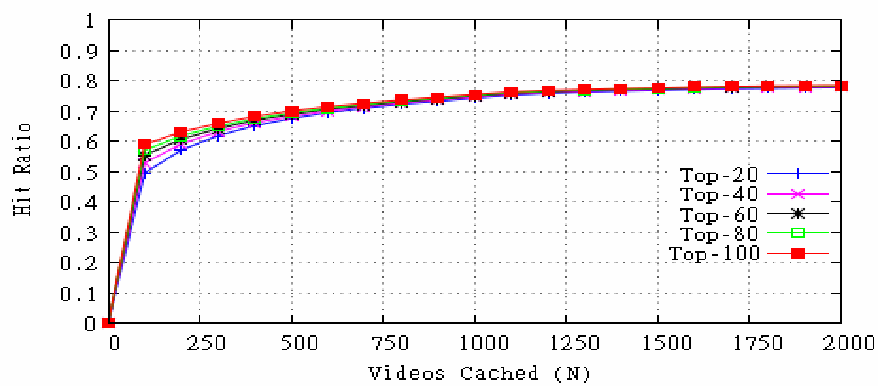


Fig. 3. Hit Ratio for combination of prefetching and caching

The combination of prefetching and caching scheme also improves the bandwidth usage as compared to prefetching-only and caching-only schemes. Prefetching-only scheme provides a maximum hit ratio of 44.2% but bandwidth consumption is very less as only 100 videos are downloaded to the cache, whereas a caching-only scheme uses more bandwidth by downloading 5767 videos to provide a higher hit ratio of 45.5% with storage space of 10GB. The combination scheme with 100 prefetchied videos and 100 cached videos will yield a hit ratio of 59% and requires 4439 videos to be downloaded where as the caching scheme of 20GB storage which offers a hit ratio 55.5% requires 4713 videos to be downloaded. The hit ratio and bandwidth savings increase in the combination scheme with increase in storage space. Thus, implementing a combined scheme of prefetching and caching works well for serving more requests from the local storage and reducing the amount of bandwidth usage in the backbone network.

5 Conclusion

In this paper, we present a measurement study of hulu traffic in a large university campus network. The analysis of the measurement data reveals that 77.69% of the video requests for hulu content are multiple requests for the same content. This is significantly higher than earlier findings on the analysis of YouTube traffic [9] where only 25% of the requested videos are requested more than once.

We analyze three different schemes, prefetching-only, caching-only and a combination of prefetching and caching, respectively. The advantage of such proxy-based distribution schemes is the fact that a viewer can access the video content faster and, since popular videos are prone to be requested multiple times, the amount of streams originating from the hulu server is reduced, resulting in a reduction of backbone bandwidth consumption. Results from our trace-based simulations show that, in the case of hulu, prefetching popular videos to the proxy is more efficient in bandwidth savings than simple caching. Prefetching the 100 most popular videos yields a hit ratio of 44.2% while a caching scheme that requires the same storage space results in a hit ratio of 45.5% with download requirement of 5767 videos. A scheme that combines prefetching and caching enhances the hit ratio by an additional 3 to 5% with less bandwidth consumption.

To the best of our knowledge, this is the first measurement-based study of hulu traffic in a large university campus network. Hulu is different than most other Internet-based services like YouTube and Netflix since it offers a variety of TV shows immediately after their broadcast on the traditional TV network. Our measurement and simulation results show that prefetching and a combined prefetching and caching approach are well suited for such a VoD service.

In future work, we plan to execute a long term measurement study to evaluate the influence of the weekly popularity of videos by the release schedule of new content and if that information can be used to further optimize the prefetching mechanism.

References

1. Hulu, http://www.hulu.com
2. Netflix, http://www.netflix.com
3. Wikipedia on Hulu, http://en.wikipedia.org/wiki/Hulu
4. Endance DAG Network Monitoring Interface, http://www.endance.com
5. Liu, W., Chou, C.T., Yang, Z., Du, X.: Popularity-wise Proxy Caching for Interactive Media Streaming. In: Proceedings of LCN Conference, Tampa, Florida (2004)
6. Sen, S., Rexford, J., Towsley, D.: Proxy Prefix Caching for Multimedia Streams. In: Proceedings of IEEE INFOCOM (1999)
7. Wu, K.-L., Yu, P.S., Wolf, J.L.: Segment-based Proxy Caching of Multimedia Streams. In: Proceedings of the 10th International Conference on World Wide Web, pp. 36–44. ACM, New York (2001)
8. Zink, M., Suh, K., Gu, Y., Kurose, J.: Watch Global, Cache Local: Youtube Network Traffic at a Campus Network – Measurements and Implications. In: Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), Santa Clara (2008)
9. Zink, M.: Scalable Video on Demand: Adaptive Internet-based Distribution. John Wiley and Sons, Ltd, Chichester (2005)