Multimed Tools Appl (2010) 46:91–118
DOI 10.1007/s11042-009-0309-5

Workload generation for YouTube

Abdolreza Abhari · Mojgan Soraya

Published online: 18 June 2009

Abstract This paper introduces a workload characterization study of the most popular short video sharing service of Web 2.0, YouTube. Based on a vast amount of data gathered in a five-month period, we analyzed characteristics of around 250,000 YouTube popular and regular videos. In particular, we collected lists of related videos for each video clip recursively and analyzed their statistical behavior. Understanding YouTube traffic and similar Web 2.0 video sharing sites is crucial to develop synthetic workload generators. Workload simulators are required for evaluating the methods addressing the problems of high bandwidth usage and scalability of Web 2.0 sites such as YouTube. The distribution models, in particular Zipf-like behavior of YouTube popular video files suggests proxy caching of YouTube popular videos can reduce network traffic and increase scalability of YouTube Web site. YouTube workload characteristics provided in this work enabled us to develop a workload generator to evaluate the effectiveness of this approach.

Keywords User generated content · YouTube · Web 2.0 · Synthetic workload generator · Video on demand · Peer-to-peer

1 Introduction

During the past 3 years networked video sharing has become a very popular web application. YouTube (http://www.youtube.com), the most successful short video sharing site, streams its videos based on user requests with about 20 million viewers daily and a total viewing time of around 10,000 years till August 2006 [6]. It has been one of the fastest

A. Abhari (✉)
Computer Science Department, Ryerson University, Toronto, ON M5B 2K3, Canada
e-mail: aabhari@scs.ryerson.ca

M. Soraya
Electrical Engineering Department, Ryerson University, Toronto, ON M5B 2K3, Canada
e-mail: msoraya@ryerson.ca

growing websites in the Internet representing a service which is different from the traditional Video-On-Demand (VoD) systems. In traditional VoD systems, content is produced by the site provider and then accessed by viewers. The content and quality of VoD applications are controlled by site owners. In contrast, YouTube enables users to participate and contribute to the site. YouTube users can upload their clips and discuss the contents by using interactive features available on the site. YouTube videos can be uploaded anytime by anyone. The quality and content of the videos vary enormously. Studying the attributes of YouTube and other video sharing sites is valuable for network traffic management. Distinct characteristics of YouTube site direct researchers to many new challenges. For example, huge popularity of YouTube has imposed a significant impact on the Internet traffic which resulted in scalability limitations for YouTube.

In this paper, we provide an extensive analysis on the characteristics of YouTube videos. Our work starts with a measurement study of YouTube traffic. During duration of five months in 2007 and 2008, we crawled YouTube site to collect the information of more than 17,000 popular videos and around 230,000 regular videos. Using this collection of data sets, we investigate YouTube videos properties such as video length and video file size. We also study meta-data features such as number of views, user rating, and number of ratings. These properties reflect the popularity and access patterns of YouTube videos.

Our YouTube measurement findings propose that using a suitable caching strategy can enhance accessing YouTube videos. The effectiveness of chosen caching scheme can be determined by developing a workload simulator. To capture workload generator properties, an analytic approach is used. Analytic workload modeling is performed with empirical measurement of various workload characteristics.

The remainder of the paper is organized as follows. Section 2 describes related works. Section 3 explains our gathering information method of YouTube videos, which is analyzed in more depth in Section 4. Section 5 discusses the implications of our results and explains the structure of developed workload generator. The simulations and results are also explained in this Section. Then Section 6 concludes the paper.

2 Related work

In this work we have focused on YouTube workload characterization as well as caching popular YouTube videos to address the YouTube performance issues such as scalability and large bandwidth demand. There has been a few research studies related to our work concerned with the analysis and characterization of YouTube short video sharing in the Internet. We found also a limited works on YouTube caching and prefetching to improve multimedia delivery performance. Since most of the works on YouTube caching are done by the same authors who worked on workload characterization we explain all the works regarding to both of these issues next. We could not find any similar work on development synthetic workload generator for YouTube in our literature review.

A YouTube traffic characterization is presented by Gill *et al.* [5]. They examine popularity and referencing characteristics, usage patterns, file properties, and transfer behaviors of YouTube videos in a campus network. The authors also analyze social networking aspect of YouTube from the edge network perspective. The edge network considered for their measurement is the University of Calgary campus network which consists of approximately 28,000 students and 5,300 faculty and staff. At an edge network, the number of users is lower than the number of global users and the edge network is physically closer to the clients requesting the content.

Gill believes that caching method could improve the end user experience and decrease network bandwidth for accessing YouTube site. However they conclude caching algorithms for traditional server are totally different from the You Tube caching because of following reasons: 1) the size of data in YouTube are much larger than traditional model. 2) The larger cache size may be required for YouTube caching to increase hit ratio. 3) Cache replacement policy should be different from traditional models of cache replacement policy that work based on recently and frequently usage of data. For YouTube additional information gathered by meta data such as user rating and content topic can also be considered for design of effective cache replacement policy. However, they did not implement or propose any caching model for You Tube.

Chattopadhyay *et al.* [3] propose a variation of MPEG Fine Grained Scalability (FGS) profile to generate the layered video representation which is suitable for multimedia progressive downloading sites such as You Tube. This approach creates layered video representation. They propose a caching scheme for the layered video representation. The proposed caching model is based on proxy caching and a new cache replacement policy which is referred to as Layered Greedy Dual Size (LGDS). LGDS works same as Greedy Dual Size (GDS) cache replacement policy; however, unlike GDS, LGDS is aware of relation between the layers and exploits this relation to achieve better cache performance. For the experiment since there is not any available trace, they use well known distributions and create 20 layered video files each of them 5 seconds duration. They compare the proposed cache replacement policy with LFU and LRU, and the result shows that the proposed caching policy outperforms conventional LRU and LFU caching policies.

Halvey *et al.* [7] provide an analysis of the social interactions on YouTube to understand community behavior. Their results show that many users do not form social networks in the online Web sharing sites like YouTube. A few users, who utilize site facilities frequently, create social networks within the site. They find that the distribution of views is not a Zipf-type distribution but the distribution of number of uploads and the number of favorite videos is a Zipf-like distribution.

Zink *et al.* [12] analyze the content distribution in YouTube and then drive a measurement study of YouTube traffic in a campus network. They monitor You Tube traffic between their campus (University of Massachusetts) and You Tube server. Three different traces are measured: First one over12 h and two traces that are the collection results from multi days and one of them (third trace) is recorded during a summer break which fewer students are in the campus. The result shows that less than 25% of all requested videos are requested more than once. They demonstrate the implications of three content delivery infrastructures: client-based local caching, P2P-based distribution, and proxy caching. The simulation results show that local caching approach improve the overall system performance. P2P-based caching shows a marginal or worse performance than the client-based caching architecture. Simulation results of proxy caching strategy exhibit an effective low-cost solution. The results of their overall simulation show that caching has more potential to decrease network traffic and reduce video access time compared to the other types of content delivery method for YouTube.

Zink's simulation on caching is based on the traces they collected for the measurement study. Since each video stream has a time stamp in the trace therefore they could use FIFO cache replacement policy for caching experiments. The results of client based caching show that even the small size of local cache can enhance the overall performance of the system. In this experiment, with 50 Mbytes client cache size, 3283 of 3899 (of the video clip which are requested more than once) were served from local cache. In proxy caching simulation, they assume that there is a proxy cache which serves the campus's access network. If the requested video clip is not located in the proxy cache, then the proxy will decide to whether store the video clip or not based on its cache replacement policy. In this experiment, the oldest video clip is removed from proxy cache to make spot for newer requested video clip. The experiment is done for proxy cache size between 100 MB and 150 GB, and the result shows that when the cache size changes from 100 MB to 1 GB the performance increase 10%, and the performance is maximum when the cache size is 100 GB. The overall result shows that the proxy cache for local access network with thousands of clients is an effective and low cost solution.

Cha *et al.* [2] consider file referencing behavior of user generated content in more details. They sample Daum UCC (http://ucc.daum.net), the most popular UGC service in Korea, and YouTube repository by using web crawler to study file referencing pattern. They focus on the popularity distribution by performing simulations and empirical validation. They observe Zipf-like behavior in the body of the popularity distribution of YouTube contents. Based on a trace-driven simulation, authors conclude caching the most popular videos can reduce server traffic. However, they do not make any assumptions about the exact location of the caches. Their target is investigation of the global cache performance from the server's point-of-view. Cha *et al.* also provide insights into P2P distribution system and find out the small number of files can benefit from P2P.

Cha simulates three different caching schemes: static finite cache in which at day zero all the long-term popular videos are stored in the cache and nothing will be added to the cache. Dynamic infinite cache in which all the video clips that requested before day zero stored in the cache and after that all of other requested videos captured by traces will be added to the cache. Hybrid finite cache which works like static cache but it has additional storage for the most popular requested video of each day. In all three schemes the full video is stored in the cache even if the clients watch half of the video clip. The result shows that 40% of the requested videos each day are different from the long-term popular videos. The static finite cache use 84% less space than dynamic infinite cache, and still manage the 75% load of the server. The hybrid finite cache is the best schema and improves the performance 10% compared to the static finite cache.

The similar work to our approach is study of Cheng *et al.* reported in [4]. They use traces crawled in a three-month period and then analyze the traces to obtain characteristics of YouTube workload. In addition, the trace data is used to investigate a caching or peer-to-peer system in order to study the benefits of such approaches. The authors suggest prefix caching [9] for YouTube videos. In this approach, proxy caches a 5 second beginning part (approximately 200 KB) of the most popular videos for each video. Finally, they conclude utilizing a delivery method based on peer-to-peer architecture for YouTube could be challenging and make the situation even worse. Although, their social network findings drives them to suggest a new approach employed by peer-to-peer technique. They propose a novel peer-to-peer system based on social network, in which peers are re-distributing the videos that they have cached. A prefetching strategy is used to prefetch the beginning part of the next video in the related social network in each peer in order to decrease the start up delay of P2P overlay. Social networks existing among YouTube videos are made by groups of related videos, if one video has been watched, another video within the group will be more likely to be accessed. Their design is based on the following principle: peers are responsible for re-distributing the videos that they have cached.

Our research is complementary to the previous ones as we compare our results with aforementioned findings. A fact that distinguishes our work from prior studies is our goal. Based on our analysis and the previous works results, we model YouTube characteristics to develop a synthetic workload generator that enables us to evaluate the performance of caching or P2P content delivery architectures for YouTube. We couldn't find any similar work for developing YouTube synthetic workload generator. In this work we used the workload generator for studying proxy cache approaches for YouTube popular files.

3 Methodology of measurement

In this paper, we focus on the high level characteristics of YouTube. For this purpose, we have crawled YouTube site in a period of five months to obtain information on YouTube videos through a combination of YouTube API [1] and software developed by us. The results represent YouTube video characteristics and user behavior.

3.1 YouTube video file properties

YouTube is a web-based service allowing the sharing of videos on the Internet. The video playback technology of YouTube is according to Adobe Flash Player. Clients can upload their own videos in different video formats like MPEG, WMV, MOV, AVI, and MP4 formats. YouTube accepts uploaded videos and converts into FLV (Adobe Flash Video) format [11]. Using FLV played a key role in the success of YouTube.

YouTube video ID is a unique 11 characters including A-Z, a-z, 0-9, -, and _ symbols and video meta-data consists of author, title, upload time, category, video length, view count, rating count, comment count, average rating, and a "related videos" list. The related videos are considered as videos with the similar description, title, keyword, or tags. They are retrieved through a hyper link of a video Web page.

3.2 Data collection strategy

In the first phase of our data gathering, popular data collection, we focused on the top 100 most viewed videos in a day and in a week to draw insights into the YouTube popular attributes. We developed software in java language to retrieve the top 100 most viewed videos of YouTube. The software establishes a connection to YouTube Web site and utilizes an API function provided by YouTube for developers to identify the top 100 videos in a day and in a week (http://www.youtube.com/api2_rest?method=youtube.videos.list_popular&dev_id=dev_id&time_range). The result of the YouTube API function is an XML file included some statistics such as the video ID, video length, average rating, rating count, view count, upload time and access time. In order to provide the video file sizes, each video ID is extracted from the XML file and sent to a server (http://cache.googlevideo.com/get_video?video_id&origin=1) which is one of the YouTube contents providers. This routine is repeated every day and every week during the data collection period to gather daily and weekly data sets respectively. In this way, we are able to collect the attributes of the popular video files which we referred to as "popular data set" based on daily or weekly duration, for further analyses.

In order to search YouTube regular clips that we name it "regular data set" we modified the crawler. Our regular data set comprises of a beginning set of top ranked list consists of "Most Discussed", "Most Viewed", "Recently Featured", and "Top Rated" video files, for the time range of "Today", "This Week", "This Month" and also "All Time". The next data sets are related videos of the previous ones. The related videos are defined as the contents linked to other videos with a similar description, title, keyword, or tag named by uploaders.

We ran the crawler between two to three times per week such that in every crawl the beginning set initials with the top ranked videos comprising of 100 to 200 videos. We collected a number of YouTube videos in a graph-like structure, where each video is a node and the top ranked videos are the first level in the graph. When video b is seen in the list of related videos of video a, then we can assume there is an edge from video a to video b. Thus, video b can be retrieved from the list of related videos of node a. A recursive method by our crawler is used to identify all related videos up to four levels. Whenever the crawler finds a video, the related video list is checked and the video is added to the list in case of being a new video. The crawler ignores any repeated videos. For each video in the list, at the first step, the crawler retrieves information from YouTube API including our required meta-data except file size. Then the crawler connects to one of the YouTube servers to extract video file sizes. It also rejects videos with incomplete meta-data information.

From the first crawling from September to November 2007 for popular data set, we provided 4,300 complete, unique, daily and weekly popular videos from 17,000 observed contents. On the second crawling between February and April 2008, the crawler traversed YouTube to four levels of the graph to obtain totaling 43,544 complete, unique regular videos from 230,000 processed contents.

4 Characteristics of YouTube videos

In our measurements, some characteristics such as category, length, size, and date added are considered as static attributes. Their values do not change in a period of data collection. Other characteristics like view count, rating count, and comment count that change from time to time are referred to as dynamic features. The dynamic information is assumed to be static in a single crawl. The summary statistics of YouTube including number of unique video files based on the highest view count, view count, average rating, video length, file size, and rating count is shown in Table 1 for regular and in Table 2 for popular video files.

In Tables 1 and 2, a is the shape parameter, b refers to the scale parameter, and coefficient of variation (COV) is a relative measure of variation among the data. It is the ratio of the standard deviation σ to the mean μ (http://en.wikipedia.org/wiki/Coefficient_of_variation) and is calculated based on formula 1:

$$\mathrm{COV} = \sigma/\mu \tag{1}$$

Low-variance distributions are recognized by COV<1 and high-variance distributions are defined with COV>1 (http://en.wikipedia.org/wiki/Coefficient_of_variation).

Our rating count analysis is based on 42,941 regular rated videos that means 98.6% of entire regular data set are rated by users. For popular data set, 100% of gathered data have rating count. We also noticed that entire regular and popular data set accounting for 100% of data contain view count meta-data value.

We observed the mean values of average rating, rating count, and view count items in regular data set are higher than the similar values in popular data set. The reason is that the beginning set of regular data set is a sample of videos from the list of top ranked videos. The next data sets are built by traversing related videos of the list of top ranked videos. Therefore, the high mean values for regular videos is a result of existence the most popular videos with large values of rating count, average rating, and view count as a starting point of the regular data collection procedure.

Table 1 Regular videos statistics

Unique IDs	43,544
Video duration (minutes)	
Mean	4.55
Median	3.91
COV	1.05
Model	Log-Logistic
	a=2.54
	b=226.54
File size (KB)	
Mean	9,809.52
Median	8,480.33
COV	0.93
Model	Gamma
	a=1.80
	b=5,441.93
Average rating	
Mean	4.56
Median	4.75
COV	0.11
Model	Weibull
	a=15.95
	b=4.73
Rating count	
Mean	1,309
Median	298
COV	3.61
Model	Weibull
	a=0.52
	b=629.24
View count	
Mean	541,564.31
Median	145,119
COV	3.18
Model	Weibull
	a=0.55
	b=290,130

In the following sections, we present a detailed analysis of the tables items by plotting their distribution models. For popular data set, we only present daily popular plots since the similar behavior for weekly popular videos is observed.

4.1 Video duration

YouTube consists of short clip videos. In our entire data set, the majority of the regular video durations are between 4 and 5 min. Figure 1 shows the histogram graph of YouTube

Table 2 Popular videos statistics

Time frame	Daily	Weekly
Unique IDs	3,605	700
Video duration (minutes)		
Mean	4.42	3.89
Median	3.41	3.11
COV	0.93	1.08
Model	Weibull	Weibull
	a=1.13	a=1.13
	b=278.03	b=246.07
File size (KB)		
Mean	10,631.41	9,269.17
Median	8,330.83	7,297.01
COV	0.93	1.11
Model	Weibull	Weibull
	a=1.144	a=1.133
	b=11,193	b=11,212
Average rating		
Mean	4.25	3.96
Median	4.60	4.32
COV	0.21	0.23
Model	Weibull	Weibull
	a=3.63	a=4.73
	b=4.44	b=4.25
Rating count		
Mean	228	802
Median	69	357
COV	2.23	1.47
Model	Weibull	Weibull
	a=0.62	a=0.75
	b=148	b=668
View count		
Mean	52,759.18	271,090.60
Median	27,382.50	190,827.50
COV	1.74	1.01
Model	Log-Logistic	Log-Logistic
	a=2.32	a=2.05
	b=30,149	b=195,492

regular video durations. Cheng [4] analysis depicted three peaks of YouTube video durations. The first peak is within one minute, which identifies YouTube as a site of very short videos. The second peak is between 3 and 4 min. This peak is seen due to the existence of a large number of videos in the Music category. The third peak is close to 10 min that is the limit of the videos duration. They found that the histogram of video

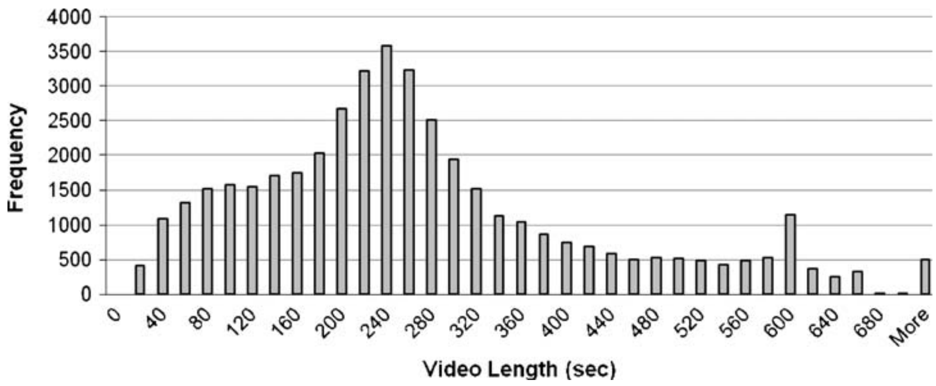


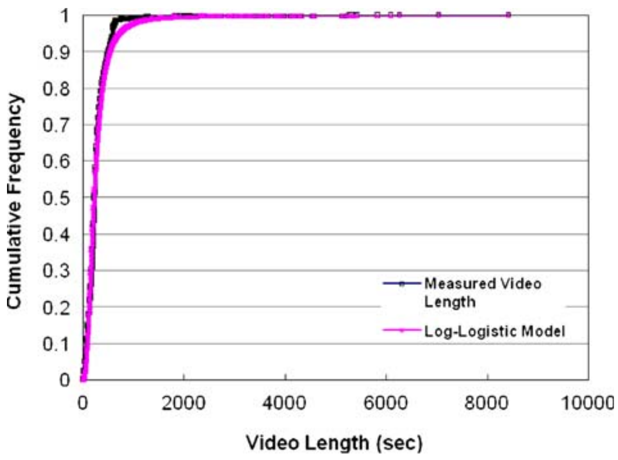
Fig. 1 Histogram of regular video durations

duration can be fit by a normal distribution. Our analysis indicates two peaks. The first one is around 4 min and the second one is close to the YouTube limitation of 10 min. Our 4 min peak may be the result of the huge volume of music videos and 10 min peak is outcome of the YouTube video duration limit. We also observed 3.4% of videos in our data set are longer than 10 min, as some authorized users have permission to upload videos longer than 10 min. Also since the length limitation was applied from March 2006 (http://youtube.com/blog), it is possible the large videos have been uploaded before that date.

Figure 2 exhibits CDF graph of video durations of YouTube regular videos. The CDF graph shows that the regular video durations can be modeled by a log-logistic distribution, whose parameters shown in Table 1.

Figure 3 shows a histogram of the popular video durations in daily and weekly data set. The mean value of daily popular video duration is 4.42 min with a median of 3.41 min. The coefficient of variation (COV) is close to 1. We also found that 56% of the daily popular videos and 64% of the weekly popular videos are between 1 and 5 min long. Our analysis indicates that durations of YouTube popular videos in our daily and weekly data sets are longer than the durations of YouTube daily and weekly popular videos found by Gill [5].

Fig. 2 CDF of regular video durations



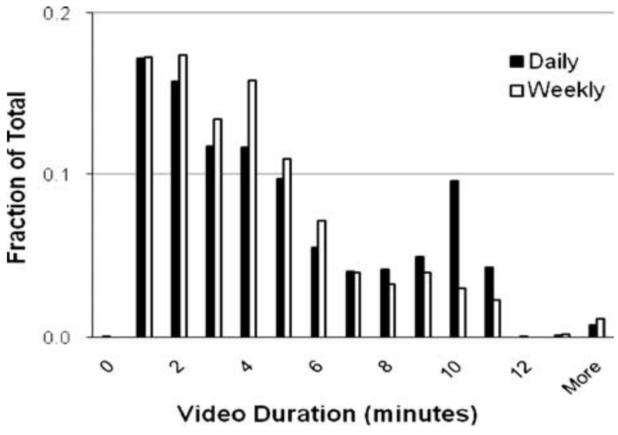


Fig. 3 Histogram of popular video durations

Gill's study had found that the median value of the durations of YouTube popular videos was around 3 min for daily popular videos and 2.2 min for weekly popular videos. The difference that we observe in our data sets is likely to be a result of different time of measuring YouTube characteristics.

Figure 4 plots CDF graph of YouTube popular video durations. The popular video duration is fitted by a Weibull distribution, whose parameters are shown in Table 1.

4.2 File size

Using the video IDs from the YouTube crawl, we retrieved 43,544 regular video file sizes. In our data, 90% of the regular videos are less than 19.2 MB and nearly 0.1% of our regular data contains video files larger than 100 MB. We calculated that the average regular video file size is about 9.8 MB whereas Cheng [4] found it as 8.4 MB. Considering the mean value that we found for regular video file size and existence of 83.4 million videos on

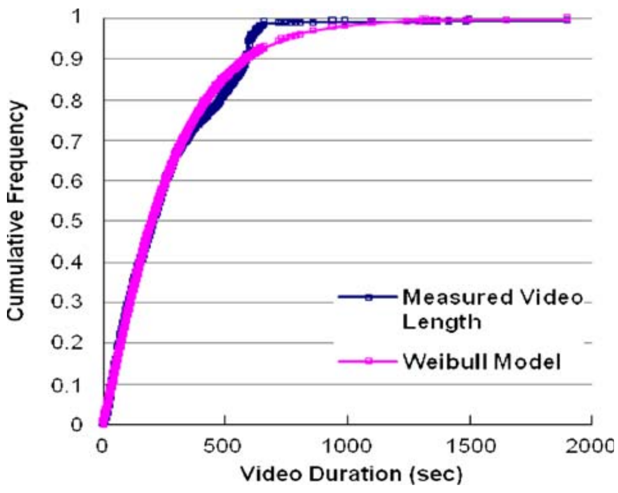


Fig. 4 CDF of daily popular video durations

YouTube servers [10], as of April 9, 2008, the required disk space to store all YouTube videos is more than 817 terabytes. For such fast growing site, a high efficient storage management must be considered. We found that the YouTube regular file size distribution can be modeled by a Gamma distribution (shown in Table 1). The CDF plot of YouTube regular video file sizes and its model are drawn in Fig. 5.

The CDF plot of the file sizes for daily popular video files is shown in Fig. 6. Since we observed the similar characteristics for weekly popular video file sizes, its CDF plot is not included in Fig. 6 to avoid cluttering.

YouTube places a limitation of 100 MB on the video file sizes (http://www.google.com/support/youtube/bin/answer.py?answer=55743&topic=10527). A small number of the videos, approximately 0.1% in our popular data set, are larger than 100 MB that shows the restriction of file size is not applied properly. In addition, there are not significant numbers of large size video files posted or accessed by users as we found that 90% of the popular videos requested by users are less than 23.5 MB. Gill's analysis [5] showed that YouTube popular video file sizes are slightly less than what we found. In their study 90% of the requested popular videos are less than 21.9 MB. These facts and the low coefficient of variation (COV) of file sizes suggest using a disk-based caching can be effective if a proxy caching system employed for YouTube popular video files.

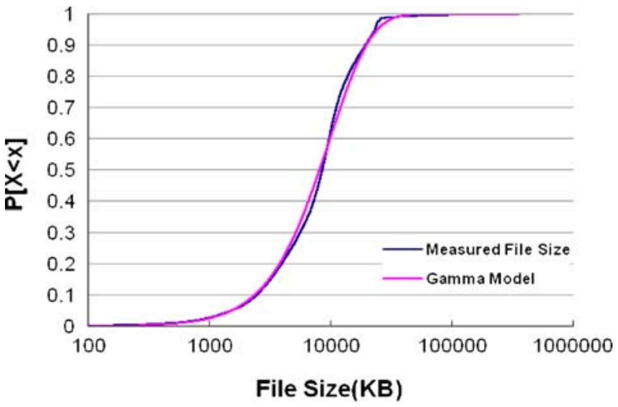
Based on our measurement, Weibull and lognormal distributions can be used for modeling the popular video file size characteristics in this study. To be able to assess how well lognormal and Weibull models fit the distributions of the popular data set, the parameters of lognormal and Weibull were measured. Also, visual observation of CDF plot, using goodness-of–fit methods as suggested in [8] and running ExpertFit (www.averill-law.com) simulation software show the Weibull distribution provides a better representation for YouTube popular video file sizes.

4.3 Average rating of videos

Average rating is the interactive feature provided by UGC sites. The average rating of a video provides a criterion of how much YouTube users liked a video clip. This user interaction attribute is used to rate videos in a scale of 0 to 5 red stars (0 showing low and 5 indicating high). Figure 7 plots the histogram of average rating for our regular data set.

Figure 8 shows the CDF plot of average rating of regular videos. We found that Weibull distribution fit better than other models for our regular data set.

Fig. 5 CDF of regular video file sizes



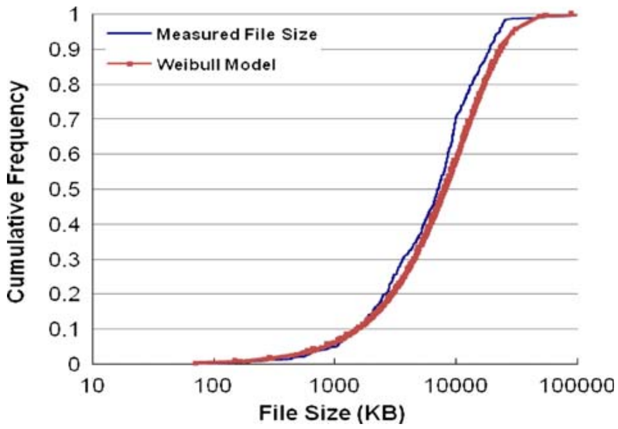


Fig. 6 CDF of daily popular video file sizes

In Fig. 9 we present a histogram of average ratings for popular video files.

The CDF model of popular video files is in Fig. 10 Weibull distribution fits our popular data set.

The regular data set graphs show over 90% of the average ratings are 4 or more. In Table 1, the mean average rating value for regular data is recorded as 4.56 with coefficient of variation (COV) of 0.11. It should be noted that the presence of high rating videos in regular files is the result of considering the top ranked video list as the starting set for collecting regular contents.

For all sets of popular videos, we also observed over 90% of the ratings have the average of 3 or higher which is close to Gill's results [5]. Based on Table 2, the mean values of average rating of daily and weekly popular videos are 4.25 and 3.96 with very little coefficient of variation (COV) of 0.21 and 0.23 in daily and weekly data sets respectively. Again, there is a very similar observation by Gill's analysis where the mean rating values are 4.20 and 3.93 and the coefficient of variation (COV) is 0.24 and 0.23 for daily and weekly popular videos respectively. YouTube contains a high volume of video files such that searching a content of interest for a viewer is difficult. Consequently, YouTube rating system helps viewers to find the high rating video among the group of their desirable YouTube videos.

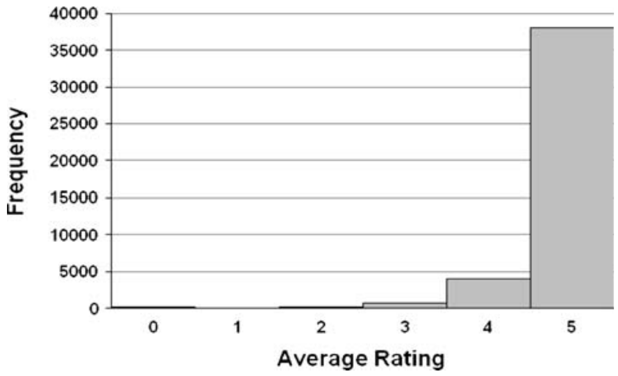


Fig. 7 Histogram of regular files average rating

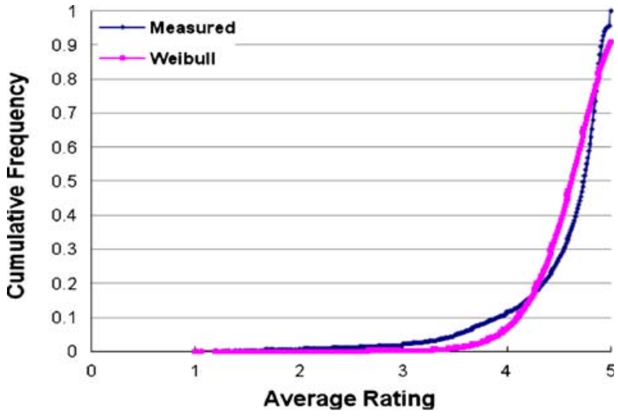


Fig. 8 CDF of regular files average rating

4.4 Rating count

Rating count meta-data for each video indicates number of users who rated that video. Tables 1 and 2 show the mean and median values of rating count are fewer than view count in our data. This fact is caused as YouTube provides the rating system just for registered users. This point demonstrates that many users do not rate a video. The CDF graphs of rating count property for regular videos in log-log scale are modeled in Fig. 11. Weibull distribution observed as the best candidate to model rating count characteristic in our regular data.

Figure 12 shows the CDF graphs of rating count feature for daily popular contents in log-log scale. The best candidate to model rating count characteristic in our daily and weekly popular data is Weibull distribution. Similar to the most of the graphs for our popular data, to avoid cluttering in Fig. 12, weekly graph is not included in the plot.

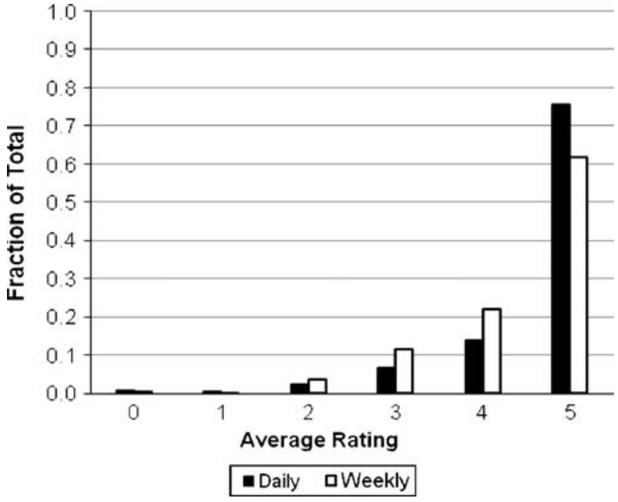


Fig. 9 Histogram of popular files average rating

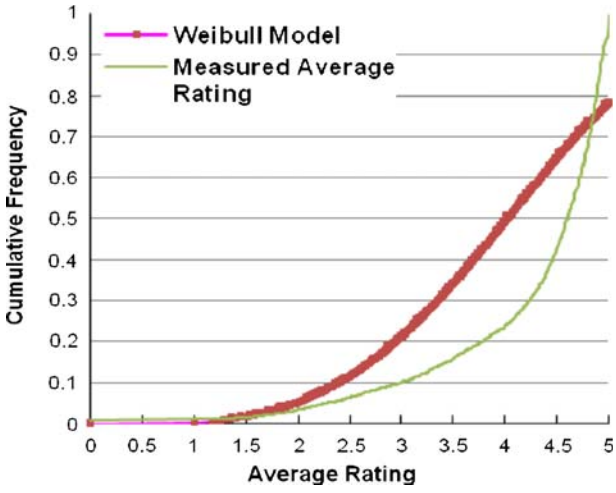


Fig. 10 CDF of daily popular files average rating

4.5 Popularity analysis of videos

One of the significant characteristic in our data sets is the number of views a video has, as it indicates the access patterns and popularity of the videos. Although, this property is a dynamic feature, we considered it to be static in our data set during the data collection period. From the obtained data, distinct videos are extracted based on the largest view count observed for each video. The distribution of popularity has an important effect on the choice of cache behavior, since popular files will tend to stay in caches. Zipf analysis in access pattern data is a method to evaluate cache efficiency. The observance of more Zipf-like behavior is an indicative of better cache performance.

A Zipf-like graph is plotted based on the rank ordered list of objects versus the frequency of the access to that object on a log-log scale and the existing of a straight line is an indication of Zipf's distribution. Figure 13 shows the view count of regular files as a function of the video rank. The plot does not indicate a Zipf distribution as it has a long tail on a log-log graph. In addition, regression analysis with the exponent ß=0.63 and the value

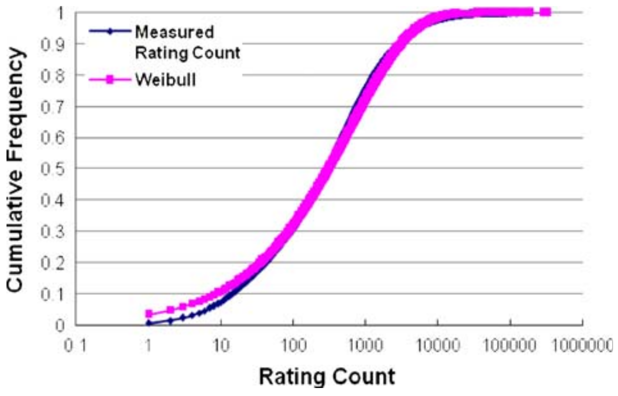


Fig. 11 CDF of regular files rating count

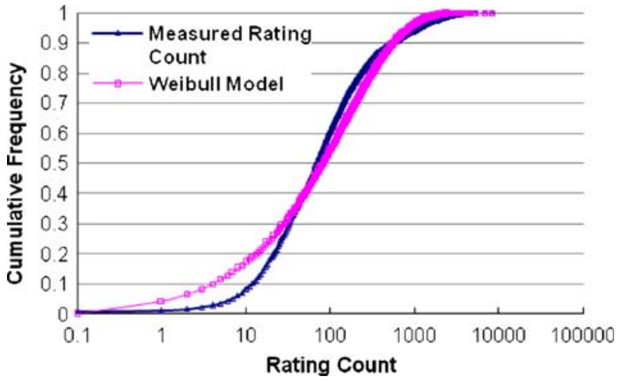


Fig. 12 CDF of daily popular files rating count

of 0.71 for goodness of fit R^2, does not represent a Zipf distribution in the popularity graph of regular videos. It follows the Weibull distribution.

This result is compatible with the findings of Cheng [4] representing video accesses on a media server which also indicate popularity does not follow Zipf's law with ß=0.54. The plot shows the head of the graph is linear but the tail sharply drops off. It suggests less popularity of regular contents at the tail which is in contrast with Zipf's law stating the curve is skewed linearly from the beginning to the end.

Based on our results, the regular files view count graph can be modeled by Weibull distribution better than other models which is shown in Fig. 14. Considering Fig. 13 and not observing Zipf-like behavior in regular videos popularity pattern, using proxy caching for the regular videos may not be an efficient solution for YouTube.

Figure 15 shows that the popularity distribution for our daily popular data set follows a Zipf-like distribution on a log-log scale. This distribution of references among video files means that some files are extremely popular, while most files have relatively few references.

Based on a regression analysis on the popular data set, the exponent ß=0.80 and goodness of fit R^2 value is 0.98. This ß value is more than the values reported by Gill [5]. Note that higher ß value means more Zipf-like behavior. Observing the ß value of 0.80, we expect to have an effective proxy caching when proxy only caches YouTube popular files.

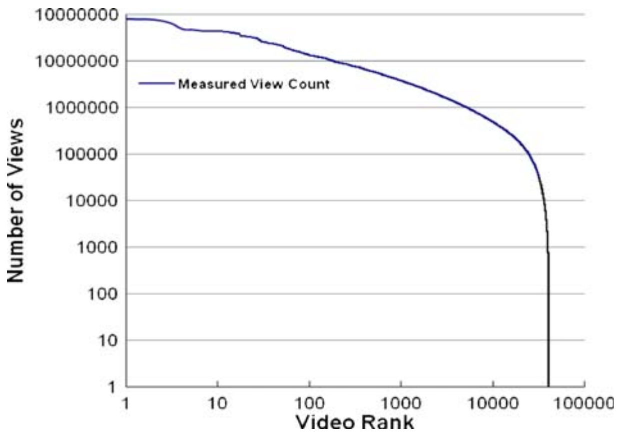


Fig. 13 Ranked video view count of regular files in log-log scale

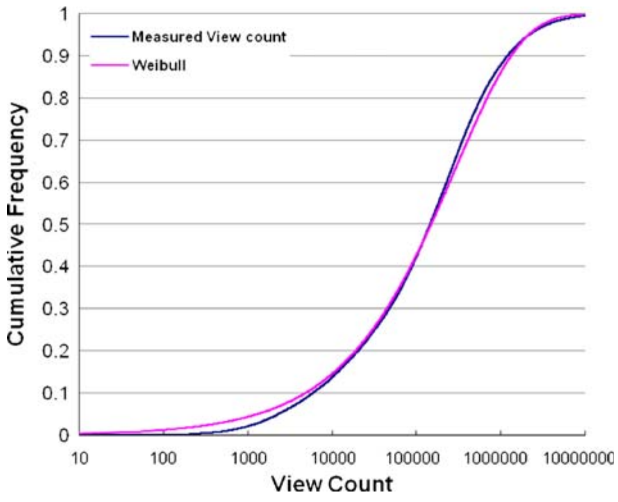


Fig. 14 CDF of regular files view count

In Fig. 15, we see a linear graph at the beginning of the plot and a tail at the high rank positions. The existence of the tail shows that there are less regular videos at the high rank position of our popular data set than the regular data set observation.

Figure 16 shows CDF graph of daily YouTube popular view count. The popular view count is modeled by a log-logistic distribution displayed in Fig. 16.

As YouTube infrastructure does not allow downloading of videos, a considerable number of video clips are requested more than once resulting in increasing the number of views. When considering the reference behavior of users, we observe more Zipf-like distribution for popular files. This characteristic indicates that caching the popular contents in proxy has the potential to decrease network traffic and bandwidth requirements for serving YouTube popular video files. We will discuss the implications on caching of our results in more detail in Section 5.

4.6 Duration of popularity

The objective of popularity duration is to determine the amount of time that a video file remains in the most popular video list. In order to investigate this feature of YouTube popular video files, we observed YouTube traffic for 5 consecutive days, hour by hour,

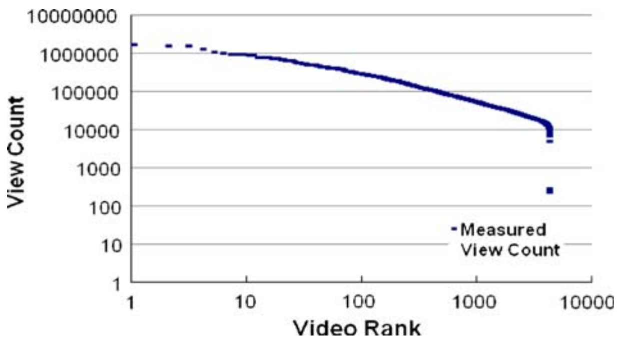


Fig. 15 Ranked view count of daily popular files in log-log scale

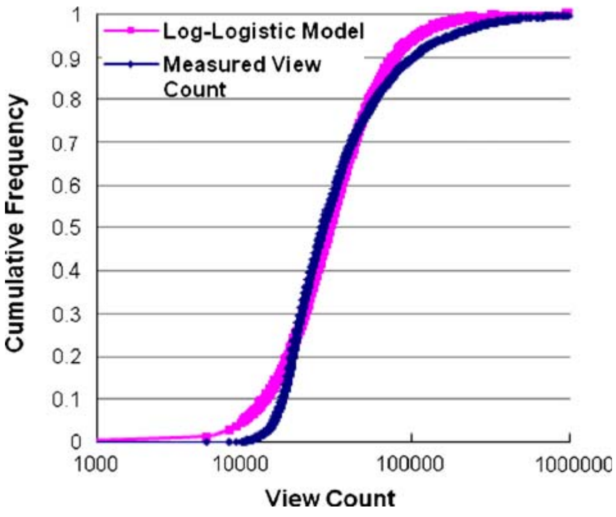


Fig. 16 CDF of daily popular files view count

from the evening of Dec.13, 2007 to Dec.17, 2007. In total we considered 10,200 most viewed videos whereas the number of unique videos is 766.

We found that the mean value of popularity duration for most viewed videos in our data set is 12.22 h with a median of 9 h. The coefficient of variation (COV) is less than 1 (COV=0.91) suggesting videos on the most popular list are not highly variable in their popularity durations. The minimum duration that a popular content stays in the popular list is 1 h and the maximum time is 41 h. Our analysis on duration of popularity suggests that the optimum time value for the cache refreshment of popular videos should be around every 12 h.

4.7 Correlation between meta-data attributes

In order to investigate the correlation between meta-data items, we calculate the correlation factors between them. For example, we examine the correlation of video length with video view count. Also we intend to see if there is any correlation between video length and video average rating, or possible correlation between video view count and video average rating. Therefore, we investigate correlation coefficient, symbolized by the letter ρ, to find out if there is any possible link between mentioned meta-data attributes.

The correlation coefficient $\rho_{X,Y}$ between two random variables X and Y with mean values E(X) and E(Y) and standard deviations σ_X and σ_Y with coefficient of variation (COV) of X and Y variables is defined (http://en.wikipedia.org/wiki/Correlation) with formula 2:

$$\rho_{X,Y} = \frac{COV(X,Y)}{\sigma_X.\sigma_Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{(E(X^2) - E^2(X))}\sqrt{(E(Y^2) - E^2(Y)}} \tag{2}$$

Table 3 lists the possible relations of different properties of YouTube regular and popular video files.

Table 3 Regular and popular correlation statistics

Characteristics	**Regular videos**		**Daily popular videos**	
	Correlation coefficient	**Relation**	**Correlation coefficient**	**Relation**
Video Length & Average Rating	0.002	Weak	0.2	Weak
View Count & Average Rating	0.048	Weak	0.18	Weak
Video Length & View Count	0.018	Weak	0.08	Weak

Based on Table 3, we conclude weak correlations exist between tabulated attributes. For example, the correlation value between video duration and view count (the third row) and the relation of video duration and average rating (the first row) reflect duration of a video has a weak correlation with its popularity. This fact proposes a weak correlation among cited features. We expect that high rating videos be more popular than low rating videos, but the low coefficient correlation of view count and average rating (the second row) does not suggest this fact. We believe some of the meta-data attributes such as view count can be used to design the cache replacement policy for popular video file caching. It means a policy which removes a video from the cache with the least view count may outperform other cache replacement policies for video caching.

5 Implications of results

5.1 Implications on caching

An efficient approach to save bandwidth and prevent user latency is caching the most used data at proxies close to clients. Existing the high number of videos with smaller size in YouTube than traditional videos, not observing a Zipf distribution in view count property of regular files, and observing a Zipf's law for popular contents, all have implications on web caching. Observation of Zipf-like behavior in popular data set suggests that proxy caching of only YouTube popular videos instead of caching of all kinds of videos can significantly increase the scalability of the server and decrease the network traffic. Caching YouTube most popular contents by cache refreshing time of every 12 h is probably the effective choice. Since a proxy receives requests from many clients it caches popular requests in the long run. However proxy caching may not be an ideal solution for regular files. We believe that YouTube can benefit from other methods to improve serving regular files that is explained in the following paragraphs.

Exploiting the concept of related videos can improve the cache efficiency at the client site for regular videos. In the other word, for a group of videos related to each other, it is very possible a viewer watches next video from the related list after viewing the previous one. Therefore, once a video is played back by a user, the related videos of the selected video can be prefetched and then cached in the client site. Cheng *et al.* [4] have also suggested that this fact can be used to design new peer-to-peer methods for short video sharing. In this work we examined the idea of proxy caching of popular videos by conduction simulation and proposed the idea of prefetching of related videos at the client sites for regular videos.

Full-object caching of videos for both caching popular videos and prefetching related videos is somehow wasting the storage and bandwidth. As for each video, the number of related videos is significant and user may watch none of the related videos. Considering this fact, the optimum number and bytes of the related videos for prefetching in the client site should be determined. Another vital issue is the cache space limits and cache replacement strategy that can be used to save the cache space. The important impact of pre-fetching is on the startup delay. Pre-fetching results in eliminating or reducing the startup delay in the client site and enhancing the playback quality. To be able to examine these approaches the first step is developing a workload generator that is discussed in Section 5.2 and next step is performing cache simulation that is explained in Section 5.3.

5.2 Workload generator characteristics

In order to design a workload generator, we analyzed workload characteristics and derived the important features of related distributions. We provided a set of probability distributions to model workload characteristics of video files on server such as file size, video length, and view count which shows popularity of individual files on the server. The workload generator can be divided into two main parts naming server workload generator and client session generator. The server part generates a collection of files corresponding to the distributional models in order to create server database. The purpose of client session generator is to simulate a user when accessing video files.

5.2.1 Server workload generator

Server Workload Generator simulates the files available on YouTube server. YouTube database is designed based on algorithm 1. After generating the videos, their related videos are created. The number of the related videos for each video is determined based on a random number between 1 and the maximum number of the related videos set by the user. This process is repeated for all videos in the related videos list.

Server workload simulator contains the form "Distributions" shown in Fig. 17 to represent distributions parameters for each video characteristics.

5.2.2 Client session generator

Client Session Generator simulates user accessing the server by selecting a video from available videos and generating an output stream.

To simulate YouTube client session, choosing data is started from the all available videos and continued until the random number generated for client session time is reached. We suppose that the first video is chosen randomly. The first selected video may have a list of related videos. We assume a user with the same probability may choose another video from all available videos or only chooses the next video from the related videos. In both of these cases (*i.e.,* whether the client session simulator chooses a video from all available videos or related videos), selection is based on the probability that is proportional to the view count field of the videos. Therefore, videos with the larger value of view counts are more likely to be selected. To explain the process of choosing next video based on view count, because of simplicity we explain the

Algorithm 1 GenerateVideos(*n, r*)

```
read $n$ as the number of videos
read $r$ as maximum number of related videos
Generate $n$ video based on distribution characteristics
for each $n$ video do
        Create_related_videos($MaxRelatedvideos \leftarrow r$)
end for
Create_related_videos($MaxRelatedvideos$)
 $k \leftarrow$ Generate a random number between 0 and $MaxRelatedvideos$
 if $k > 0$
        for each $i \le k$ related video do
                $j \leftarrow$ Generate a random number between 1 and $n$
                The address of $i^{th}$ related video $\leftarrow$ The address of $j^{th}$ video
        end for
end if
```

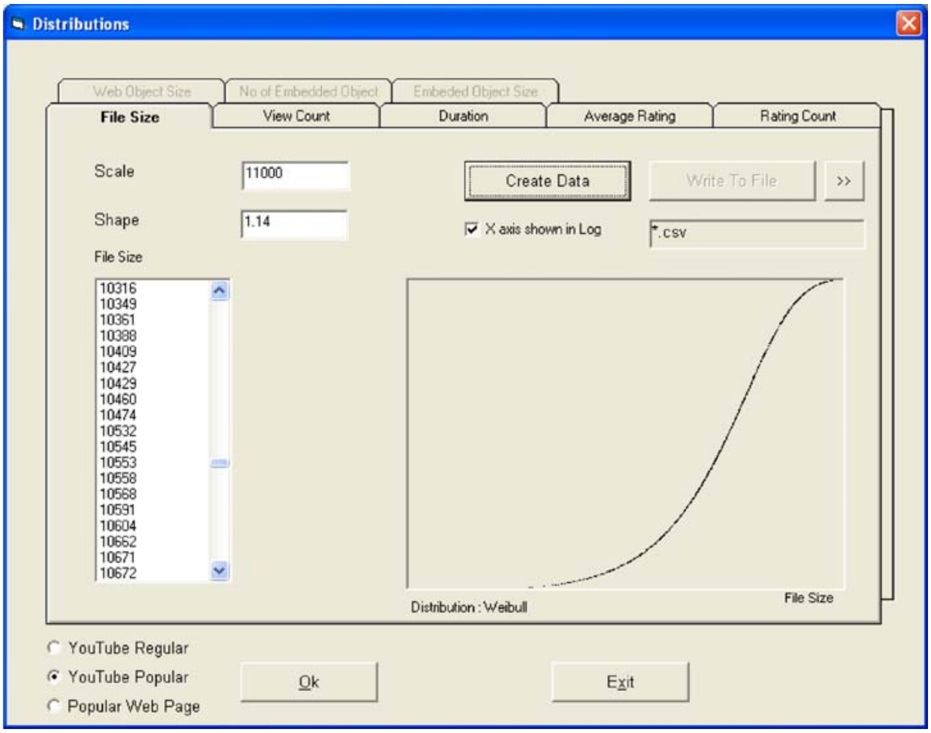
scenario of choosing a related video from a related video list. First we define following terms and formulas:

r	number of related videos for each video
VC	View Count of each video
S	total view counts of related videos for each video

$$s = \sum_{i=1}^{r} vc_i \tag{3}$$

To select the next video out of the *r* related videos, a random number *m* which would be between 1 and *s* (Eq. 3), the total number of the related videos view counts, is created. Then the application will find the video ($j(1 \le j \le r)$) with the total view count ($\sum_{i=1}^{j} vc_i$) greater than m if the total view count of the previous video (j-1), $\sum_{i=1}^{j-1} vc_i$, is less than m. It means when m has a value between $\sum_{i=1}^{j-1} vc_i$ and $\sum_{i=1}^{j} vc_i$, the jth related video will be the next selected video. The expression 4 explains how related video is chosen.

$$\begin{cases} \sum_{i=1}^{j-1} vc_i < m \le \sum_{i=1}^{j} vc_i \quad , & 1 < j \le r, 0 < m \le s \\ 1 \le m \le vc_j \quad , & j = 1 \end{cases} \tag{4}$$



Scale: 11000

Shape: 1.14

File Size:

10316
10349
10361
10388
10409
10427
10429
10460
10474
10532
10545
10553
10558
10568
10591
10604
10662
10671
10672

[✓] X axis shown in Log

*.csv

Distribution : Weibull

File Size

YouTube Regular [✓ YouTube Popular] Popular Web Page

Fig. 17 Distribution form

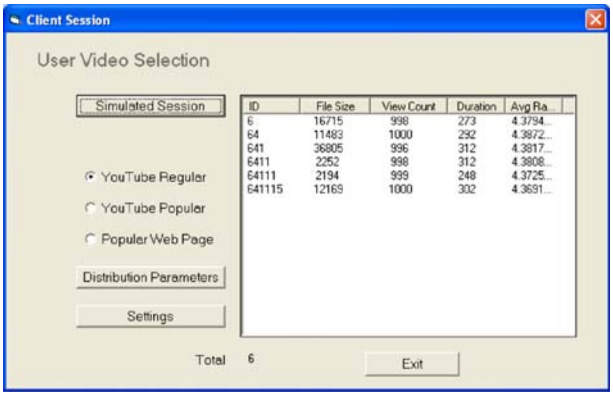
For selecting a video from all available videos in server exactly same approach is used in which expression 4 applies to all of the available videos. This approach guaranties the selection of the next video is based on a probability that is proportional to the video view count.

Client session simulator continues selecting videos based on the above strategy until the client session time is reached.

Client simulator contains “Client session” form shown in Fig. 18 that shows user selection videos when we limit the simulator to only chooses from the related videos of previous video. For conducting the simulations in next section as other simulation parameters we used a default setting which is selection of the next video with equal probability from the related videos of current video (if related videos exist) or another video from all available videos in the server.

5.3 Simulations and results

Based on observation of Zipf-like distribution in popular videos we decided to simulate proxy caching of popular YouTube videos. We selected a proxy as a location of cache since proxy is located between server and client and acts on behalf of many clients therefore deals with popular requests. In real system a proxy or a group of cooperative proxies can be used for caching popular YouTube videos upon receiving them from the server and forwarding them to the requested clients. First we used the workload generator to generate a log containing the requests of clients to the popular YouTube



Client Session

User Video Selection

Simulated Session

(● YouTube Regular

○ YouTube Popular

○ Popular Web Page

ID	File Size	View Count	Duration	Avg Ra...
6	16715	998	273	4.3794...
64	11483	1000	292	4.3872...
641	36805	996	312	4.3817...
6411	2252	998	312	4.3808...
64111	2194	999	248	4.3725...
641115	12169	1000	302	4.3691...

Total 6

Fig. 18 Client session form

videos. Then we added a time stamp to each client accessing video in the log. The timing information is generated by assuming a Poisson distribution arrival for the clients. The differences of the Poisson random numbers are taken and put as the next access time after the current access. The next access time was calculated after finishing watching the video and each subsequent access continues from that point. We assumed there are no incomplete video streaming by the clients in this simulation. Finally we generated several logs each contained 250,000 requests for the 50,000 distinct popular videos. It should be considered that some of the videos were requested multiple times, and others just once.

For the implementation of the proxy cache simulator Java language was used. Each log used as an input file for our proxy cache simulator to perform the experiments. We used the default parameters of distributions as shown in Table 2, to generate the logs capturing requests to popular videos. We conducted a number of experiments to study the effectiveness of proxy caching with changing workload parameters. In the first experiment we generated two traces each of them contained 250,000 requests to 50,000 daily and weekly popular videos. Figure 19 shows the performance of proxy

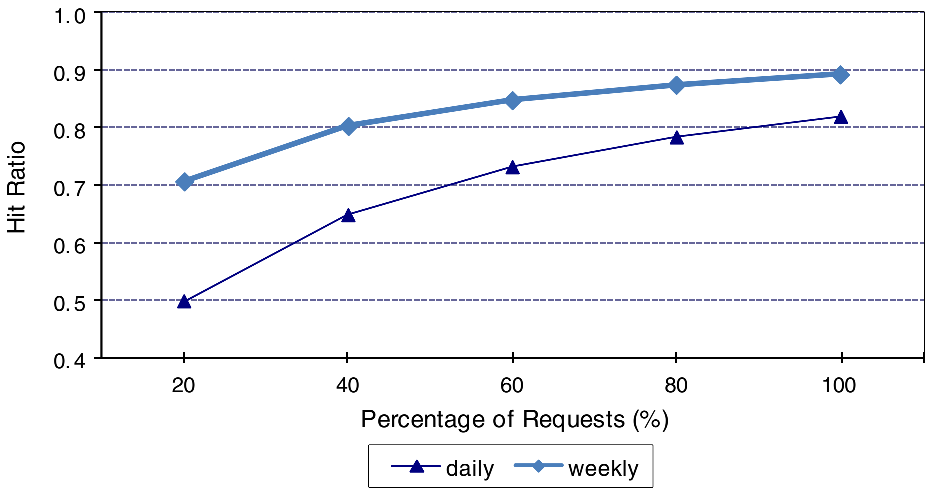


Fig. 19 Cache performance with infinite cache size for popular videos

caching when the number of requests change. To show cache performance hit ratio was used. Hit ratio is the ratio of the number cache hits and the total number of cache requests. A hit is said to have occurred when the referenced file is in the proxy cache. The higher hit ratio is better because it means more documents were served from the proxy cache therefore more server load reduction was achieved. As Fig. 19 shows with more percentage of requests both daily and weekly traces gain more hit ratio. We used infinite cache size to be able to exclude the effect of cache replacement policy in this experiment. Figure 19 also indicates that caching popular weekly videos that has the longer range of popularity has better performance compared to popular daily videos. Table 2 shows popular weekly distribution model has larger shape parameter *a* compared to popular daily distribution model. Shape parameter *a* represents the skewness parameter of view counts that is modeled by Log-Logistic distribution. To study the relation between shape parameter *a* and cache effectiveness and number of requests we generated five different popular daily traces when shape parameter *a* change. Figure 20 shows the smaller shape parameter *a* results higher hit ratio that we observed in comparison of popular daily and weekly cache performance shown in Fig. 19.

Figure 20 shows less value for *a* means more concentration on accessing smaller numbers of videos which results better cache performance. Note that we can verify the effect of changing shape parameter *a* visually by the graphical representation of the graph provided in our simulator as shown in Fig. 17. We also observed that increasing the number of requests in the trace increases the cache performance.

To study the effect of cache size on cache performance of our proposed approach we conducted another set of experiments by changing cache size. We generated two traces with default parameters and 250,000 requests for daily and weekly popular videos. The results of

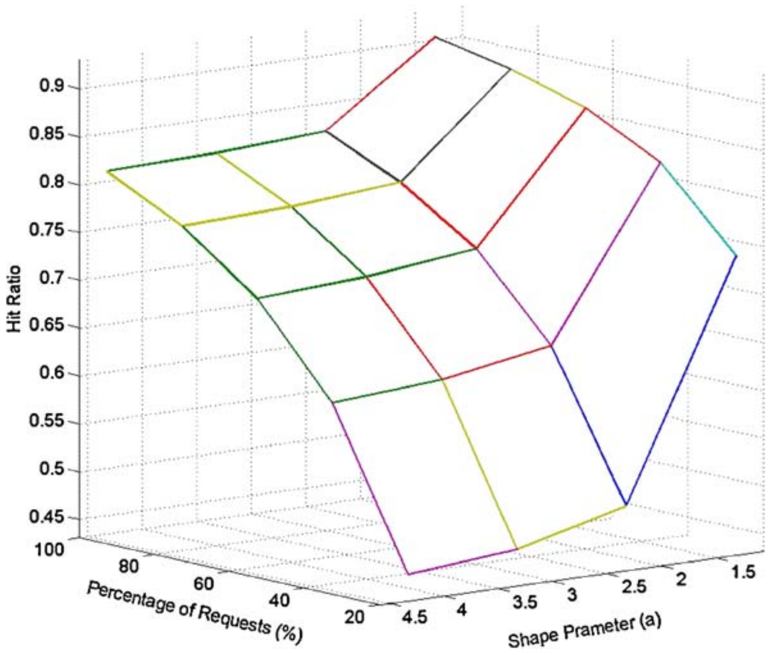


Fig. 20 Cache performance with infinite cache size for popular daily videos

cache performance simulation are shown in Figs. 21 and 22. We used Least Recently Used (LRU) approach as a cache replacement policy since it is a common cache replacement policy that is used by popular proxy applications such as Squid. LRU removes the least recently used files from the proxy if it is needed. We assumed a proxy/s employed a disk based caching with the cache size in the range of 1 GB to 10000 GB. Also since YouTube videos has a limitation in maximum files size, proxy caches a complete video files. For cache performance metrics plus hit ratio we used byte hit ratio as well. Byte hit ratio is the number of bytes that the proxy cache served directly by providing from its cache as a percent of the total number of bytes for all requests. It corresponds to the ratio of the sum of the sizes of the requested files found in the cache, and the sum of the sizes of all the requested files. Byte hit ratio can be used to show traffic reduction in the sever side. Therefore more byte hit ratio means better cache performance.

The results show that hit ratio and byte hit ratio achieved by proxy caching and LRU policy are in the range of 12% to 89% for different cache sizes. Figures 21 and 22 show that the graphs of each performance metric start to flatten off from cache sizes greater than 1000 GB. This can be explained by the fact that the total size of requested files in our simulated trace is less than 1000 GB. Achieving the average of 50% hit ratio and 50% byte hit ratio when employing 100 GB of cache size shows that caching popular video files can provide substantial performance improvement. This means by employing a dedicated single proxy cache with 100 GB disk in a local accessing network (*e.g.,* a university campus) that generates around 250,000 requests to YouTube in a given period of time, the traffic and load on the YouTube server generated by this network can be potentially reduced in half. This means the video playback time and download time in the user side in this network can be significantly reduced.

Using cooperative proxies with large storage is common approach in the higher levels of network such as root proxies and national caches and can achieved the maximum performance shown in Figures 20 and 21. However dedicating a proxy to cache only YouTube videos may not be a common method in local network that employed proxy caching. Usually a proxy with a popular application such as Squid is used for a local network to cache all the web

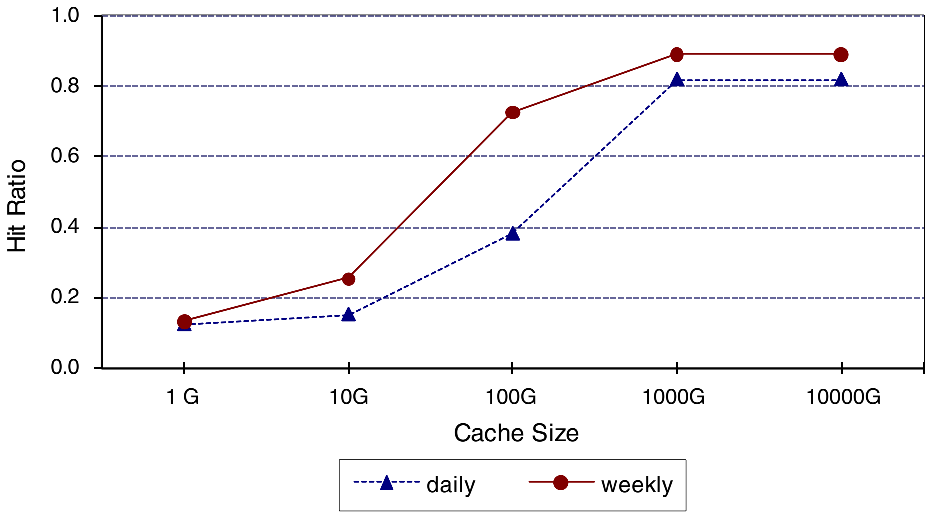


Fig. 21 Proxy caching of youtube popular videos with total caching and variable cache size

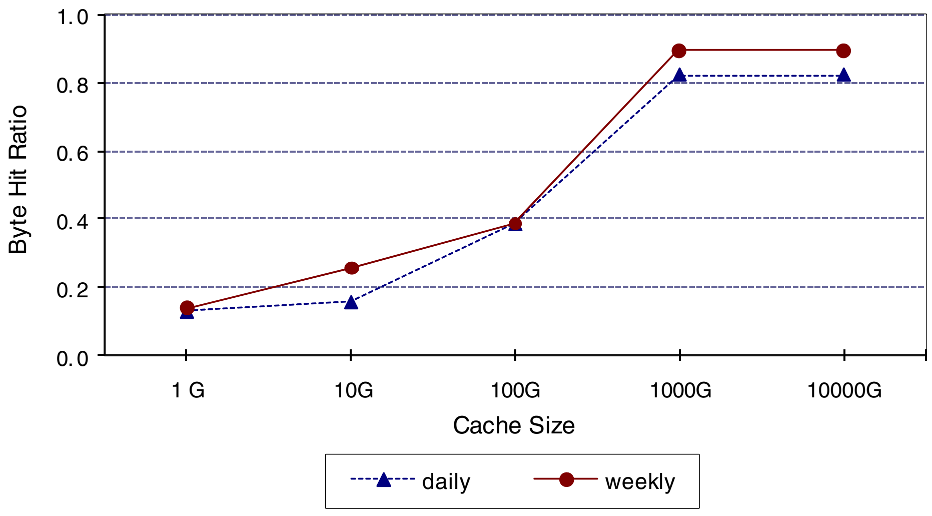


Fig. 22 Proxy caching of youtube popular videos with total caching and variable cache size

objects with different types in the same cache storage. Since YouTube video files are larger than other web objects such as text and image files it is not possible to use total caching of videos together with image and text objects. Therefore, we performed another set of experiments with partial caching of YouTube videos to study effectiveness of our approach if it used by common proxy application such as Squid for a local network. Similar to Cheng's approach [4] we only cache 5 second of each video that is required for buffering purpose. We generated the new popular traces with 250,000 requests with all default parameters. Figures 23 and 24 show the results of cache performance with partial caching.

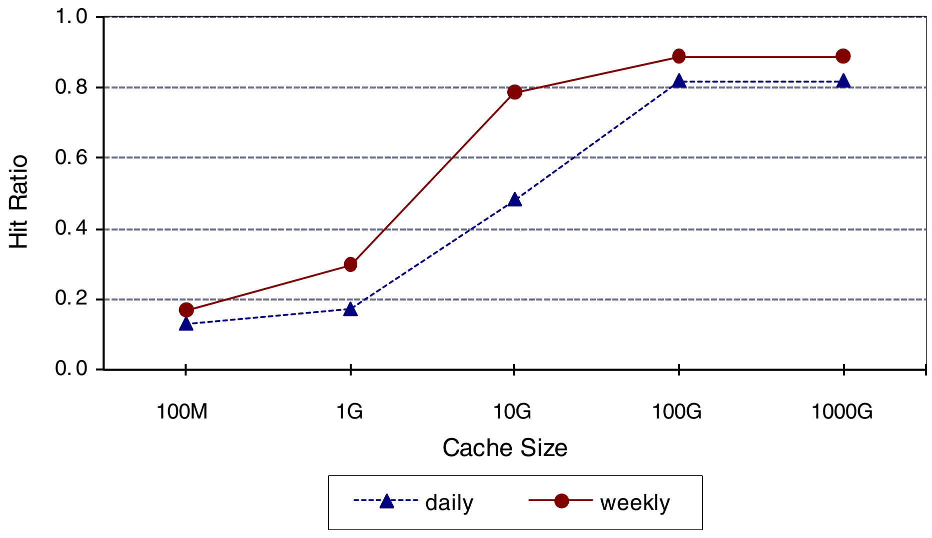


Fig. 23 Proxy caching of youtube popular videos with partial caching and variable cache size

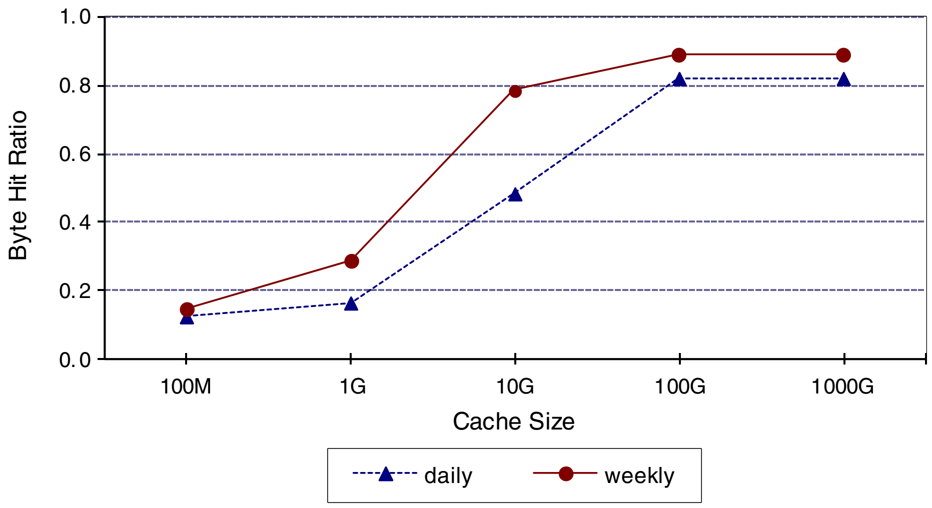


Fig. 24 Proxy caching of youtube popular videos with partial caching and variable cache size

The results of Figures 23 and 24 show that for getting the average of 50% cache improvement only 10 GB of proxy disk space is needed when doing partial caching of video files in our simulation environment. Allocation of this space in the proxy cache applications such as Squid that uses LRU policy is a simple approach and results improvement in video playback time while watching YouTube videos.

6 Conclusion

We have presented an analysis of the characteristics of one of the most popular short video sharing site, YouTube. Through investigating an extensive amount of data gathered in five months, we have represented YouTube popularity distribution and access pattern. These characteristics introduce new opportunities to improve the performance of short video sharing web sites by proposing a novel caching model.

Based on our findings, caching the most popular videos in the proxy along with prefetching the related videos in the client site can reduce the client access time and start up delay in watching video. To estimate and simulate how different caching methods perform when changing workload parameters, a simulator as a tool is required. The simulator generates synthetic workload with the same characteristics as real YouTube popular and regular videos. The main contribution of this work is presenting essential elements required for development of a realistic workload generator for short video sharing sites such as YouTube. Based on these elements, we implemented a workload generator to be able to measure the performance of our proposed caching strategy. To demonstrate the usefulness of the simulator we simulated several scenarios with variable workload parameters when caching popular YouTube files in a proxy cache. The achieved result supports our observation of effectiveness of proxy caching for YouTube popular videos. These simulation results are especially important for web proxy cache administrators.

The workload simulator presented in this work is not limited to the performance measurement of only the caching techniques. Instead, this synthetic workload generator can be used for performance measurement of a broad range of multimedia delivery architectures with different structures such as client server and peer-to-peer systems that are designed to address the high bandwidth consumption and scalability problems of the Web 2.0 video sharing sites such as YouTube. The next direction of this research is to build a framework with more simulator tools to determine the effectiveness of using peer to peer architecture for YouTube.

Acknowledgements We would like to thank the anonymous reviewers of this paper for their suggestions.

References

1. API Documentation (YouTube). http://youtube.com/dev docs.
2. Cha M, Kwak H, Rodriguez P, Ahn Y, and Moon S (2007) "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System," *In Proceedings of the 7th ACM SIGCOMM conference on Internet Measurement,* pp. 1-14, San Diego, USA.
3. Chattopadhyay S, Ramaswamy L, and Bhandarkar SM (2007) "A Framework for Encoding and Caching of Video of Quality Adaptive Progressive Download", *In Proceedings of the 15th international conference on Multimedia*, Germany.
4. Cheng X, Dale C, and Liu J (2007) "Understanding the Characteristics of Internet Short Video Sharing: YouTube as a Case Study," Technical Report arXiv: 0707.3670v1 [cs.NI], Cornell University, arXiv e-prints.
5. Gill P, Arlitt M, Li Z and Mahanti A, (2007) "YouTube Traffic Characterization: A View From the Edge," *In Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 15-28, San Diego, USA.
6. Gomes L (2006) "Will All of Us Get Our 15 Min On a YouTube Video?" Wall Street Journal.
7. Halvey M and Keane M (2007) "Exploring Social Dynamics in Online Media Sharing," *In Proceedings of the 16th international conference on World Wide Web,* Banff, Canada.
8. Law AM, Kenton WD (2000) Simulation Modeling and Analysis, 3rd edn. McGraw-Hill, Boston, pp 292–402
9. Sen S, Rexford J, Towsley D (1999) "Proxy Prefix Caching for Multimedia Streams." *In Proceedings of the 18th IEEE Conference on Computer Communications (INFOCOM'99)*, Volume 3, pp. 1310-1319, New York, NY, USA.
10. The Wall Street Journal (from Wikipedia). http://en.wikipedia.org/wiki/The_Wall_Street_Journal
11. YouTube: Video Format (from Wikipedia). http://en.wikipedia.org/wiki/Youtube#Video format
12. Zink M Suh K and Kurose J (2008) "Watch Global, Cache Local: YouTube Network Traffic at a Campus Network - Measurement and Implications," *In Proceedings of ACM/SPIE MMCN '08 conference*, Volume 6818, pp. 5-13 San Jose, USA.

Abdolreza Abhari is an associate Professor in the Department of Computer Science at Ryerson University in Toronto, Canada where he has worked for past 6 years. He obtained his Ph.D. degree in computer science from Carleton University, Ottawa, Canada in 2003. He received his M.Sc. degree in software engineering in 1992 and B. Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran in 1989. He worked as a computer system consultant for industrial business segments from 1992 to 1998 in Iran, Netherland, Germany and Canada. Dr. Abhari's research interests are in the areas of distributed systems, Web multimedia systems, Internet technology and web caching, performance measurement and workload analysis. He is a member of ACM and IEEE Computer Society.

Mojgan Soraya is recently graduated with M.A.S degree from Department of Electrical and Computer Engineering, Ryerson University, Toronto, Canada. Her thesis title is “web multimedia file characterization”. She has obtained her B.Sc. degree in Computer Engineering from Esfahan University. Before her Master’s study she has been working as a software engineer in several industrial segments for more than 5 years. Her research interests are in the areas of multimedia workload characterization, performance evaluation, modeling and simulation.