World Wide Web (2015) 18:1523–1540
DOI 10.1007/s11280-014-0318-9

Pareto-based cache replacement for YouTube

Ming-Chang Lee · Fang-Yie Leu · Ying-ping Chen

Received: 28 July 2014 / Revised: 25 November 2014 /
Accepted: 22 December 2014 / Published online: 21 March 2015

Abstract Recently, YouTube, which plays diverse video programs for worldwide users, has been one of the most attractive social-networking systems. YouTube employs a distributed memory caching system called Memcached to cache videos, and utilizes the Least Recently Used algorithm (LRU for short) to evict the least recently watched video when Memcached runs out of space. However, LRU may cause a high miss count, which is the number of times that a video requested by users cannot be found in Memcached. This might not only increase network overhead, but also cause a poor service quality for YouTube since those videos need to be retrieved from the remote back-end database. To solve these problems, in this paper, we classify videos into popular and unpopular videos and propose two cache replacement algorithms based on the Pareto principle. One is Pareto-based Least Frequently Used algorithm (PLFU for short), and the other is Pareto-based Least Recently Used algorithm (PLRU for short). The two algorithms always keep several top popular videos of each video category in Memcached to reduce miss count. However, when Memcached has insufficient space to hold a video requested by a user, PLFU and PLRU repeatedly evicts an unpopular video from Memcached based on LFU and LRU so as to hold the video. Our simulation results based on a real-world YouTube trace show that PLFU performs the best among all tested algorithms in terms of miss count and video-retrieval time. The results also indicate that when PLRU is used for a longer time, it provides the second best performance.

Keywords YouTube · Memcached · Pareto principle · Cache replacement · Least recently used · Least frequently used

M.-C. Lee · Y.-p. Chen
Department of Computer Science, National Chiao Tung University, Hsinchu City, Taiwan

M.-C. Lee
e-mail: mingchang1109@gmail.com

Y.-p. Chen
e-mail: ypchen@cs.nctu.edu.tw

F.-Y. Leu (✉)
Department of Computer Science, TungHai University, Taichung City, Taiwan
e-mail: leufy@thu.edu.tw

1 Introduction

In recent years, video traffic has been prominent on the Internet. More than 2 billion YouTube videos are accessed each day, and a huge amount of new videos are uploaded every day [23]. Labovitz et al. [14] claimed that today 15~25% of Inter-Autonomous System traffic results from video delivery. According to the statistics shown in [3], YouTube is not only the world's largest user-generated content site [26], but also the third most popular website in the world. In fact, social media such as YouTube faces a much greater challenge because of the huge demands of resources, including storage space and network bandwidth [9].

Currently, YouTube uses Memcached [27], which is a distributed memory caching system widely used by cloud and web service delivery companies, to cache videos so as to reduce the number of times for remotely retrieving a video requested by a user from the back-end database of YouTube. Memcached utilizes the Least Recently Used (LRU for short) algorithm [18, 27] as its cache replacement approach. When the space of Memcached is insufficient to keep a video *V*, YouTube make a room for *V* by evicting a video that has not been accessed for the longest time from Memcached. We call this substituted video *S*. The above replacement might increase video-retrieval time and impact the service quality of YouTube if *V* is watched rarely and *S* after being evicted is accessed again by users, implying that *S* needs to be replicated to Memcached again.

To solve the above problems, in this paper, we propose two cache replacement algorithms for YouTube. One is Pareto-based Least Frequently Used algorithm (PLFU for short), and the other is Pareto-based Least Recently Used algorithm (PLRU for short). PLFU and PLRU follow the Pareto principle (also known as the 80/20 rule, Power laws, or Zipf's law) [19] to keep popular videos (i.e., videos that are frequently accessed by users) in Memcached, rather than choosing them as the substituted candidates. The purpose is to reduce miss count, which in this paper is defined as the number of times that a video requested by users cannot be found in Memcached. When Memcached is full, PLFU and PLRU employ different ways to evict a video. The former evicts an unpopular video that has the least value of increased access count from Memcached, whereas the latter evicts an unpopular video that is the least recently accessed from Memcached. To evaluate and compare PLFU and PLRU with the other two known cache replacement algorithms, including LRU [6], and the Least Frequently Used algorithm (LFU for short) [6], we use the real YouTube trace collected by Cheng et al. [8] to conduct extensive simulation. The simulation results demonstrate that PLFU outperforms the other three algorithms in terms of miss count and video-retrieval time. The results also show that when PLRU is used for a longer time, its miss count and video-retrieval time will be less than those of LFU and LRU.

The rest of this paper is organized as follows. Section 2 introduces the background and related work of this study. In Section 3, we describe how PLFU and PLRU determine popular and unpopular videos, and then detail how PLFU and PLRU maintain Memcached. Section 4 evaluates and compares the performances of all tested algorithms. Section 5 concludes this paper and shows our future work.

2 Background and related work

In this section, we first outline the background about YouTube, Memcached, LFU, and LRU. Then we describe the related work of this study.

2.1 Background

2.1.1 YouTube

YouTube was established in 2005 and bought by Google in 2006. Currently, YouTube is the most popular video-sharing website around the world [8]. When a user wants to upload a video to YouTube, he/she can select 1 of 12 categories to define the video. Table 1 illustrates the distribution of all the YouTube video categories analyzed by [8]. We can see that "Music" is the most popular category among all the categories, and "Entertainment" is the second popular category.

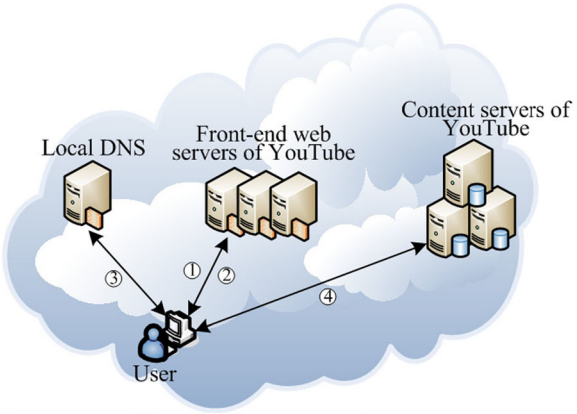
On the other hand, when a user wants to watch a video on YouTube, the corresponding procedure is as follows (see Figure 1). First, the user either directly watches the video by using the exact video page URL or searches the video at www.youtube.com. Once the user selects a video, the front-end web server will reply the user with a HTML page carrying the video information and the name of a content server for the video. Next, the local domain name server resolves the name of the content server into the corresponding IP address. Finally, the user can use the IP address to access and watch the video [2, 23].

2.1.2 Memcached

Memcached is a general-purpose distributed memory caching system developed by Danga Interactive for a social networking service called LiveJournal [17]. Generally, Memcached is used to accelerate the processing speed of dynamic database-driven websites by caching data in random-access memory. Memcached has been widely adopted by cloud and web service delivery companies, such as Wikipedia [18], Facebook [21], Twitter [24], and YouTube [12], to reduce the latency of sending web data to users and to ease the demands on databases and computational servers. In YouTube, Memcached is used to hold videos for serving video requests from users, and it can effectively accelerate the video service of YouTube by reducing the chance of remotely retrieving the requested videos from the back-end database of YouTube.

Table 1 The distribution of all the YouTube video categories [8]

Category	Percentage
Music	22.9 %
Entertainment	17.8 %
Comedy	12.1 %
Sports	9.5 %
Film & Animation	8.3 %
People & Blogs	7.5 %
Gadgets & Games	7.4 %
News & Politics	4.4 %
Autos & Vehicles	2.6 %
Travel & Places	2.2 %
Howto & DIY	2.0 %
Pets & Animals	1.9 %



①User sends a video request to YouTube at www.youtube.com
②Front-end web server replies the user with the video information and the name of a content server
③Local DNS resolves the name of the content server into the corresponding IP address
④User uses this IP address to access and watch the video

Figure 1 The procedure for a user to access and watch a video on YouTube [23]

2.1.3 LFU and LRU

LFU [6] and LRU [6] are two cache replacement algorithms, which are widely used in many areas of data management [15, 20]. If a cache has insufficient space to hold a file, LFU removes the least frequently used file (i.e., a file that has the lowest use frequency) from the cache. However, different from LFU, LRU removes the least recently used file (i.e., a file that has the longest time period since the time it was last used).

2.2 Related work

To study the popularity life cycle of videos, Cha et al. [7] collected and analyzed a large access trace from YouTube and Daum UCC, which is the most popular user-generated content (UGC) service in Korea [10]. The authors claimed that on both YouTube and Daum UCC, 10 % of the top popular videos result in near 80 % of total video access. Their analyses also showed that the popularity distribution properties of YouTube and Daum UCC follow Pareto Principle (or 80/20 rule) [19]. The authors then proposed three caching schemes. The first is a static finite cache, which is used to store those videos that have long lasted popularity. Any new popular videos will not be stored in this cache. The second is a dynamic infinite cache, in which all videos are stored at the very beginning, and any video accessed afterward will also be stored in this cache. In other words, the space of the cache is unlimited. The third is a hybrid finite cache, which is similar to that of the static finite cache, but it provides extra space to store the most popular video per day. However, the first scheme cannot adapt the population variation, the second scheme is not practical in the real world, and the third scheme did not address how to evict a video when the extra space is full.

Cheng et al. [8] collected YouTube datasets for a 4-month period in early 2007 and a 5-month period in middle 2008. Based on these datasets, the authors conducted a systematic measurement study on the statistics of YouTube videos, but they did not analyze the miss count and video-retrieval time of current cache replacement algorithms for YouTube.

Abhari and Soraya in [1] studied the traffic of YouTube and concluded that the data-access patterns of YouTube videos follow the Zipf-like distribution [5]. The authors implemented a workload generator to measure the performance of their proposed caching strategy on

YouTube, and evaluated the hit ratio and byte hit ratio of the LRU cache replacement policy on different cache sizes. Different from [1], in this paper, we provide two Pareto-based cache replacement algorithms for YouTube to not only cache videos but also to evict videos when Memcached is full.

In our previous work [16], we introduced PLFU and PLRU and studied their performances in terms of miss count. However, the corresponding impact on video-retrieval time was not analyzed. In fact, the video-retrieval time incurred by a cache replacement algorithm is a very important metric for YouTube since it represents the service quality of YouTube. Hence, in this paper, we further formally derive a metric called video-retrieval time ratio, and evaluate the corresponding results of all tested cache replacement algorithms.

3 The proposed cache replacement algorithms

In this section, we first illustrate how PLFU and PLRU determine popular videos and unpopular videos. Then we introduce how PLFU and PLRU conduct a cache replacement and maintain Memcached.

3.1 Popular and unpopular videos determination

In this study, PLFU and PLRU use the Pareto principle (or called the 80/20 rule), which is widely utilized to describe the skewness in distribution [7], to determine popular and unpopular videos.

Algorithm 1 illustrates the popular-video determination algorithm (PVD for short), which is used by both PLFU and PLRU to determine popular videos for each of the 12 video categories on YouTube. Let C_k be the k-th video category, k=1,2,…,12. Let $V_{k,i}$ be a video with an unique ID i in C_k. Whenever a fixed time period (e.g., a week) elapses, PVD calculates how many times that each video of each category is watched/accessed during this period. Let $N_{k,i}$ be the access count of $V_{k,i}$ in this time period. Then PVD calculates the increased access count of $V_{k,i}$, denoted by $I_{k,i}$, by using Eq. (1).

$$I_{k,i} = N_{k,i} - N'_{k,i} \tag{1}$$

where $N'_{k,i}$ is the access count of $V_{k,i}$ in the previous one time period. If $V_{k,i}$ is a new video, $N'_{k,i}$ must be zero since $V_{k,i}$ does not exist in the previous time period. Otherwise, $N'_{k,i} \geq 0$. For example, if $N_{k,i}$=500 and $N'_{k,i}$=300, it means that the access count of $V_{k,i}$ increases by 200 times because $I_{k,i}$=200.

Next, PVD sorts all videos of C_k based on their increased access counts in a descending order. Let N_v be the total number of videos of C_k in the time period. Then PVD considers the top 20 % of the N_v videos as popular videos, and treats the remaining videos as unpopular videos. In other words, the total number of videos that are considered as popular videos, denoted by P, is

$$P = \lfloor N_v \cdot 0.2 \rfloor \tag{2}$$

The total number of videos that are treated as unpopular videos, denoted by U, is therefore

$$U = N_v - P \tag{3}$$

After that, PVD proceeds according to the following cases:

1. If any of those P popular videos has been already in Memcached, the video will still be kept in Memcached.
2. If any of those P popular videos is not in Memcached yet, PVD requests either PLFU or PLRU to put the video into Memcached based on which one of PLFU and PLRU are employed.
3. If any of those U unpopular videos has been already in Memcached, the video will be treated as an eviction candidate such that it could be evicted when Memcached has insufficient space.

The popular-video determination algorithm:
Input: All videos on YouTube;
Output: The decision about popular videos and unpopular videos;
Procedure:

```
1:  while a predefined time period is up{
2:    for each category C_k{
3:      for each video V_{k,i} in C_k{
4:        Calculate the access count of V_{k,i}, denoted by N_{k,i};
5:        Calculate I_{k,i} of V_{k,i} by invoking Eq. (1);}
6:      Sort all videos of C_k based on their increased access counts in a descending
7       order;
8:      Calculate P and U by using Eqs. (2) and (3), respectively;
9:      Treat the first P videos of all the videos in C_k as popular videos.
10:     Treat the remaining videos (i.e., the U videos) as unpopular videos;
11:     for each of the P popular videos {
12:       if the video is currently in Memcached {
13:         Do nothing;}
14:       else {
15:         If PLFU is used {
16:           Request Algorithm 2 to put this video into Memcached;}
17:         If PLRU is used {
18:           Request Algorithm 3 to put this video into Memcached;}}
19:     for each of the U unpopular videos that is currently in Memcached{
20:       Treated the video as an eviction candidate; }}}
```

Algorithm 1. The popular-video determination (PVD) algorithm.

3.2 PLFU and PLRU

Algorithm 2 shows the algorithm of PLFU. Let N_{HIT} be the total number of videos that are requested by users and can be found in Memcached, and let N_{MISS} be the total number of videos that are requested by users but cannot be found in Memcached. In other words, N_{HIT} represents the hit count, whereas N_{MISS} indicates the miss count.

Whenever receiving a request from a user to access a video V, PLFU increases the access count of V by one, and then checks whether V is in Memcached or not. If yes (see line 4), it means that the user can directly retrieve V from Memcached. Hence, PLFU increases N_{HIT} by one.

However, if V is not in Memcached (see line 7), PLFU increases N_{MISS} by one. In this case, V will be further retrieved from the remote back-end database and kept into Memcached so as

to enable the subsequent users to quickly access V. To achieve this, PLFU checks whether Memcached has sufficient space to store V or not. If the answer is positive (see lines 8 to 9), PLFU directly stores V into Memcached after retrieving V from the remote back-end database. Otherwise, PLFU employs the following three conditions to repeatedly remove a video, denoted by S, from Memcached until V can be accommodated by Memcached (see lines 11 to 14).

1. S is in Memcached.
2. S belongs to one of the unpopular videos.
3. S has the least value of increased access count.

Let us see an example. Suppose that the music category has only five videos: V_1, V_2, V_3, V_4, and V_5. Table 2 lists the access count of each video in the 3rd and 4th weeks. The fourth column of Table 2 shows the increased account count of each video from the 3rd to 4th week. Based on the PVD algorithm, the top 20% of videos of each category will be considered as popular videos. Hence, V_1 is a popular video, and the rest four videos are unpopular videos. When Memcached has insufficient space to accommodate another video, PLFU will first remove V_4 from Memcached since V_4 is the one with the least increased access count among all the unpopular videos (i.e., V_2, V_3, V_4, and V_5).

The algorithm of PLFU
Input: A video request for accessing V;
Output: A replacement decision;
Procedure:

```
1:    Let N_HIT = 0 and N_MISS = 0;
2:    while receiving a video request for accessing V{
3:      Increase the access count of V;
4:      if V is in Memcached{
5:        N_HIT++;}
6:      else{
7:        N_MISS++;
8:        if Memcached has sufficient space to hold V{
9:            Store V into Memcached;}
10:       else{
11:           while Memcached has insufficient space to hold V{
12:               Select a video S' from Memcached where S' has the least access count
13:               among all the unpopular videos in Memcached;
14:           Remove S' from Memcached; }
15:       Store V into Memcached;}}}
```

Algorithm 2. The algorithm of PLFU.

Algorithm 3 shows the algorithm of PLRU, which is similar to the PLFU algorithm. The only differences are as follows: For each video, PLRU records the last time point that the video was watched. When Memcached has insufficient space to hold V, PLRU uses the following three conditions to repeatedly remove a video, denoted by S', from Memcached until V can be accommodated by Memcached.

Table 2 An example showing the access count of five videos in two consecutive weeks

Video ID	Access count in the 3rd week	Access count in the 4th week	The increased access count	Last watched time point
V_1	500	3200	2700	1 h ago
V_2	2500	2800	300	4 days ago
V_3	100	1000	900	3 h ago
V_4	50	95	45	3 days ago
V_5	125	600	475	12 h ago

1. S' is in Memcached.
2. S' belongs to one of the unpopular videos.
3. S' is the least recently watched video, implying that S' has not been watched for the longest time.

Let use the same example listed in Table 2 to explain PLRU. Note that the last column of Table 2 shows the last time point that each video was watched. Due to the fact that PLRU also uses the PVD algorithm to determine popular videos and unpopular videos, so V_1 is still a popular video, and the remaining four videos are unpopular videos. However, when Memcached has insufficient space to accommodate another video, PLRU will first remove V_2 from Memcached since V_2 has not been watched for the longest time among all the unpopular videos.

The algorithm of PLRU
Input: A video request for accessing V;
Output: A replacement decision;
Procedure:

```
1:  Let N_HIT = 0 and N_MISS = 0;
2:  while receiving a video request for accessing V {
3:    Set the access time point of V into current time;
4:    if V is in Memcached{
5:      N_HIT++;}
6:    else{
7:      N_MISS++;
8:      if Memcached has sufficient space to hold V {
9:          Store V into Memcached;}
10:     else{
11:         while Memcached has insufficient space to hold V {
12:             Select a video S' from Memcached where S' is the least recently watched
13:             video among all the unpopular videos in Memcached;
14:           Remove S' from Memcached; }
15:         Store V into Memcached;}}}
```

Algorithm 3. The algorithm of PLRU.

4 Experimental results

In this section, we evaluate and compare PLFU and PLRU with another two known cache replacement algorithms, i.e., LRU [6] and LFU [6], based on the YouTube access trace crawled

by Cheng et al. [8]. The entire trace covers 21 weeks from April to September in 2008, and total number of videos collected in the trace is 161,085. The trace provides detailed information about every video, including video ID, uploader, category, the number of cumulative access count, etc. However, the arrival time of every video request was not included in the trace. In order to replay the trace to conduct our evaluation, we assumed that the arrival time of all video requests follows a Poisson distribution [13], and each video request is a discrete and independent event. Then we generated the arrival time of each video requests based on the Poisson distribution.

In addition, to reduce the experimental complexity and meanwhile achieve a fairness comparison, we focused on the datasets of the top two categories: Music and Entertainment. The total number of music videos and entertainment videos in the dataset were 27,843 and 42,763, respectively. Each video had been accessed for several thousands of times. Such a huge amount of video requests resulted in that our simulator ran out of its memory. Therefore, we randomly selected 7000 videos from the 27,843 music videos to be our first test dataset, denoted by D_1. Similarly, we randomly chose 9078 videos from the 42,763 entertainment videos to be our second test dataset, denoted by D_2.

In our experiment, the back-end database held all the 7000 music videos and 9078 entertainment videos. However, Memcached could not hold all these videos simultaneously. We use Eq. (4) to determine the maximum number of videos of D_f that Memcached can hold at the same time, denoted by M_f, where f=1 or 2.

$$M_f = \lfloor N_f \cdot R \rfloor \tag{4}$$

in which N_f is the total number of videos in D_f, and R is a factor controlling the size of Memcached for D_f, $0<R<1$. For example, if R=1/4, then M_1=1750 (=7000·1/4) and M_2= 2269 (=9078·1/4). It means that Memcached can hold at most 1750 music videos and 2269 entertainment videos simultaneously.

To achieve a comprehensive evaluation, we created four scenarios by combining the two test datasets and two values of R (i.e., R=1/4 and R=1/3). The purpose is to see how different values of R (i.e., different available Memcached sizes) impact the performances of all tested algorithms.

Table 3 lists the details of the four scenarios, in which the values of M_f were calculated by Eq. (4). Both PLFU and PLRU use the Pareto 80/20 rule to determine popular videos. Hence, when the music dataset is tested, the 20% of the 7000 videos that have the largest value of increased access counts will be considered as popular videos. Hence, the total number of popular videos is 1400 (=7000·0.2). On the other hand, when the entertainment dataset is tested, the total number of popular videos is 1815 (=9078·0.2).

Table 3 The details of the four scenarios

Scenario	Dataset	N_f	R	M_f
1	Music	7000	1/4	1750
2	Entertainment	9078	1/4	2269
3	Music	7000	1/3	2333
4	Entertainment	9078	1/3	3026

Two metrics are considered in the experiment. One is the average miss count per week, and the other is the Video-Retrieval Time ratio (VRT ratio for short) per week. The VRT ratio shows how much video-retrieval time can be reduced by a cache replacement algorithm per week. Let N_{total} be the total number of videos requested by users during a week. Let S_j be the size of the j-th requested videos, $j=1,2,\ldots,N_{total}$. Let N_{Hit} and N_{Miss} be respectively the total hit count and miss count during the week, implying that $N_{Hit}+N_{Miss}=N_{total}$. Let S_a be the size of a-th hit video, $a=1,2,\ldots,N_{Hit}$. Let S_b be the size of the b-th miss video, $b=1,2,\ldots,N_{Miss}$. Then the corresponding VRT ratio can be calculated as follows.

$$\text{VRT ratio} = \frac{\sum_{a=1}^{N_{Hit}} \frac{S_a}{BW_{Mem}} + \sum_{b=1}^{N_{Miss}} \frac{S_b}{BW_{DB}}}{\sum_{j=1}^{N_{total}} \frac{S_j}{BW_{DB}}} \tag{5}$$

where BW_{Mem} and BW_{DB} are the bandwidths for retrieving a video from Memcached and the remote back-end database, respectively. Generally, $BW_{Mem} \ll BW_{DB}$. In fact, the denominator is the total video-retrieval time in the worst case, i.e., all the N_{total} videos need to be retrieved from the remote back-end database. It is clear that a high N_{Hit} leads to a low VRT ratio since the total video-retrieval time reduces. It is also clear that a lower VRT ratio indicates the better performance of a cache replacement algorithm.

In the following, we show the average miss count and average VRT ratio of each tested algorithm in the four abovementioned scenarios. Note that in each scenario, we performed each algorithm on the 21 weeks of YouTube access traces for ten times. Each time has its own video arrival time.

4.1 Scenario 1: the music dataset and $R=1/4$

Figure 2 shows the average miss counts of all tested algorithms in scenario 1. The corresponding standard deviations during the 21 weeks are listed in Table 4. No matter which algorithm was tested, the corresponding miss counts reduced when time went by. However, we can see that PLFU outperformed the other three algorithms since it led to the lowest average miss count. The average miss count of PLFU is about 0.739, 0.643, 0.836 times that of LFU, LRU, PLRU, respectively, during the 21 weeks. The reasons are two-fold. First, PLFU uses the increased access count of a video during a week to determine the popularity of videos, rather than using the accumulative access count of a video. Hence, PLFU can accurately distinguish popular videos and unpopular videos in every week. Second, PLFU always keeps the top 20% of popular videos in Memcached and removes only unpopular videos from Memcached when Memcached has insufficient space to hold a video. Consequently, the resulting miss count of PLFU significantly reduced.

When PLRU was tested (see the circle curve), we can see that its average miss count was higher than PLFU and LFU before the 6th week. After that, its average miss count apparently reduced. Even though PLRU cannot beat PLFU, but it performed much better than LFU in terms of average miss count. The main reason is that PLRU never evicts those top 20% of popular videos from Memcached when Memcached ran out of space to hold another video.

When LRU was evaluated (see the square curve), it almost performed the worst among all tested algorithms. We can see that it caused the highest miss count before the 19th week, implying that simply employing LRU cannot help YouTube to significantly reduce the miss count.

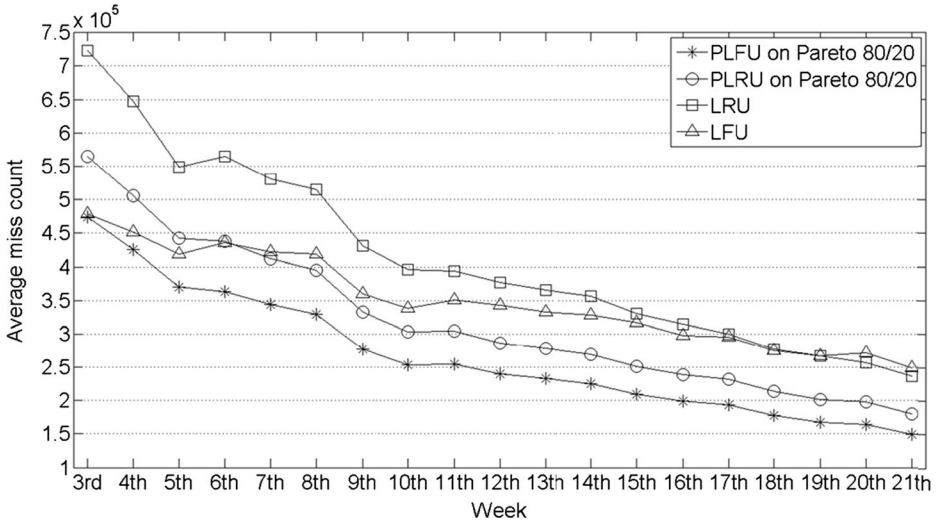


Figure 2 The average miss counts of all tested algorithms in scenario 1

When the LFU was tested (see the triangle curve), its average miss count was lower than those of PLRU and LRU before the 6th weeks. However, it was similar or even higher than those of PLRU and LRU afterward. The key reason is that LFU always evicts a video that has the least accumulative access count from Memcached. In fact, the evicted video might be a video that is newly uploaded and frequently watched during a week. We call this property a New-Popular-Video preference eviction (NPV-preference eviction for short). In other words, LFU tends to evict a new popular video, rather than a video that was uploaded long time ago and has a very high access count. The NPV-preference eviction gradually impacts the performance of LFU. We can see that the miss count of LFU is the highest among all algorithms in the 20th and 21th weeks.

Based on the results shown in Figure 2, Figure 3 illustrates the corresponding average VRT-ratio results of all tested algorithms. The related standard deviations are listed in Table 5. Due to having the smallest miss count, PLFU led to the lowest VRT ratio among all algorithms. In other words, PLFU consumed the least amount of time for retrieving videos from the back-end database.

Due to employing the same Pareto principle of PLFU to determine popular and unpopular videos, PLRU overall provides the second lowest VRT ratio (i.e., the second best performance in terms of the VRT ratio) among all tested algorithm. When LRU was evaluated, we can see its VRT ratio reduce as time went by. However, because of its high miss count during the 21 weeks (please see Figure 2), LRU's VRT ratio was almost the highest among all algorithms.

On the other hand, because of the abovementioned NPV-preference eviction, LFU was unable to reduce its VRT ratio. In fact, its average VRT ratio fluctuated during the

Table 4 The standard deviation of the average miss counts of all tested algorithms

Algorithm	Standard deviation
PLFU	37.7~626.6
PLRU	54.1~730
LRU	107.9~1283
LFU	57.5~665.5

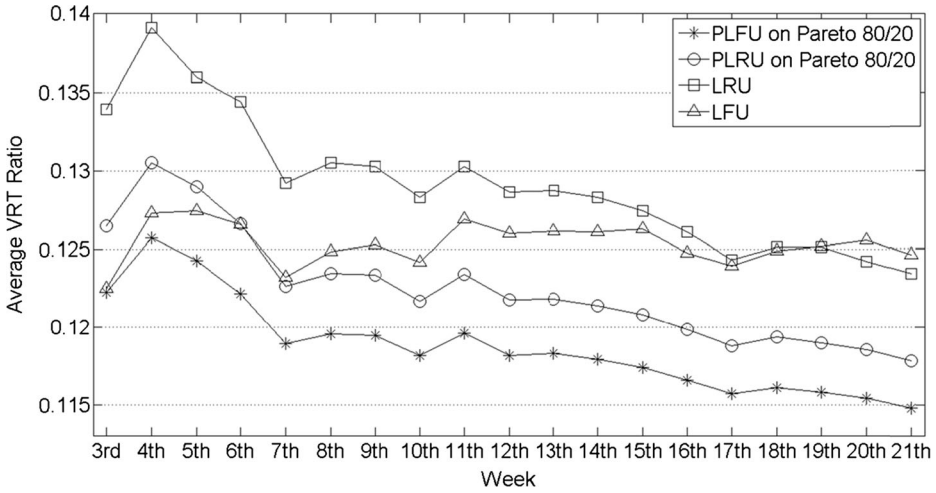


Figure 3 The average VRT ratios of all tested algorithms in scenario 1

21 weeks, implying that employing LFU for YouTube is unsuitable. Based on our results, we can infer that the VRT ratio of LFU will keep fluctuate overtime.

4.2 Scenario 2: the entertainment dataset and R=1/4

Figure 4 shows the average miss counts of all tested algorithms in scenario 2. The corresponding standard deviations are similar to Table 4, so they were not depicted to save paper space. We can see that the trends of all curves in Figure 4 are very similar to those in Figure 2. PLFU still performed the best among all tested algorithms since it had the smallest average miss count. The average miss count of PLFU is about 0.731, 0.639, 0.831 times that of LFU, LRU, PLRU, respectively, during the 21 weeks. When LFU was employed, it still performed well at first, but it was worse than the others at the end because of the NPV-preference eviction. The rest two algorithms (i.e., PLRU and LRU) have similar miss-count trends in scenario 2 as they were in scenario 1. Based on the above results, we can see that the type of dataset is not a factor to influence the performance of these algorithms. In other words, the similar performance results will also appear when the other ten video categories of YouTube, i.e., Comedy, Sports, Film & Animation, People & Blogs, Gadgets & Games, News & Politics, Autos & Vehicles, Travel & Places, Howto & DIY, and Pets & Animals [8], were tested.

Table 5 The standard deviation of the average VRT ratios of all tested algorithms

Algorithm	Standard deviation
PLFU	$7\times10^{-5}\sim1\times10^{-4}$
PLRU	$8.5\times10^{-5}\sim1.1\times10^{-4}$
LRU	$9.1\times10^{-5}\sim1.7\times10^{-4}$
LFU	$8.1\times10^{-5}\sim1.5\times10^{-4}$

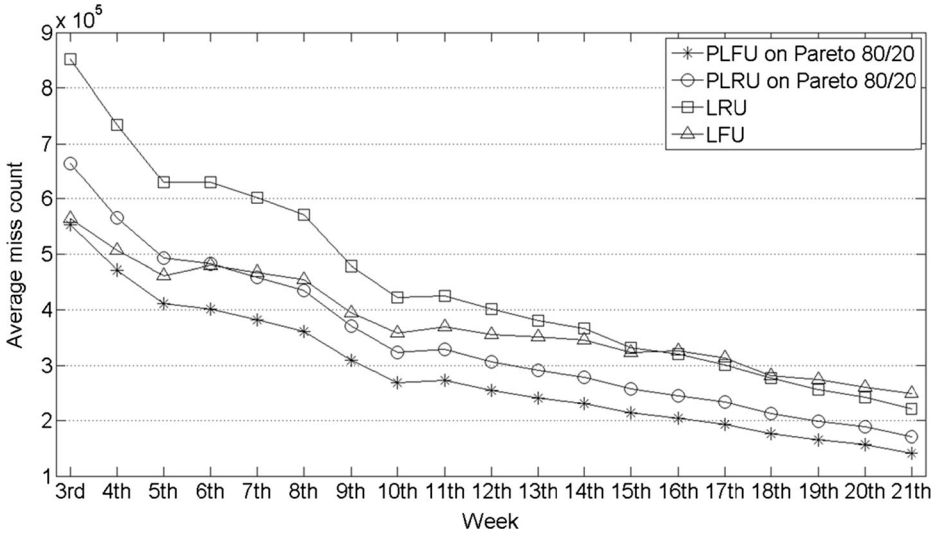


Figure 4 The average miss counts of all tested algorithms in scenario 2

Figure 5 shows the corresponding average VRT-ratio results of all algorithms in scenario 2. The corresponding standard deviations are similar to Table 5, so they were not presented to save paper space. Overall, employing PLFU, PLRU, and LRU can reduce the VRT ratio, but we still can see that PLFU led to the lowest VRT ratio among all algorithms. On the other hand, utilizing LFU still cannot decrease the VRT ratio. The reason is the same, i.e., LFU favors to evict a new popular video from Memcached, rather than an old popular video.

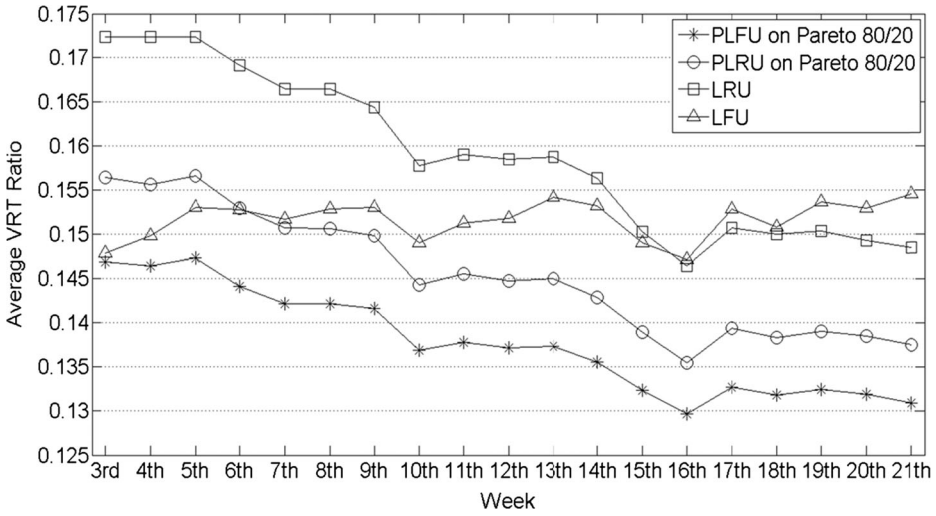


Figure 5 The average VRT ratios of all tested algorithms in scenario 2

4.3 Scenario 3: the music dataset and R=1/3

In this subsection, we show how the four algorithms perform in scenario 3. Different from scenario 1, Memcached in this scenario can hold more music videos because R=1/3. Figure 6 depicts the corresponding average miss counts of all tested algorithms. The corresponding standard deviations are similar to Table 4. By comparing all miss-count curves with those showing in Figure 2, we can see that each tested algorithm in scenario 3 led to a much lower miss count since the available space of Memcached in scenario 3 was larger than that in scenario 1. In other words, in scenario 3, more videos can be held by Memcached, so each algorithm can have more chance to retrieve a video from Memcached. Nevertheless, Figure 6 still shows PLFU outperforms the other three algorithms in terms of miss count. The average miss count of PLFU is about 0.697, 0.637, 0.699 times that of LFU, LRU, PLRU, respectively, during the 21 weeks. The results imply that the available size of Memcached did not change the fact that PLFU is the best one among the four algorithms.

Due to the NPV-preference eviction, LFU was unable to significantly reduce its miss count, causing that LFU performed the worst after the 16th week. On the other hand, we can see that the miss-count curve of PLRU is still lower than that of LRU, implying that employing the Pareto principle can effectively lower the miss count of PLRU when the size of Memcached increases since PLRU always evicts the least recently watched video from Memcached.

Figure 7 illustrates the average VRT ratios of all algorithms in scenario 3, which correspond to the results shown in Figure 6. The corresponding standard deviations are similar to Table 5. The total time spent by PLFU to retrieve videos from the back-end database is still the least. Comparing all curves in Figure 7 with those shown in Figure 3, we can see that all tested algorithms had a lower VRT ratio in scenario 3, implying that increasing the size of Memcached can reduce the VRT ratio of all the algorithms.

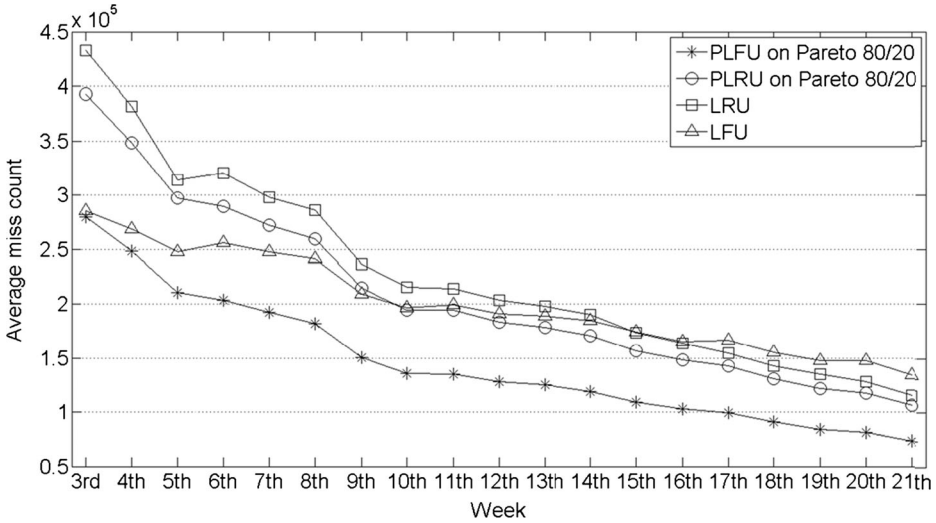


Figure 6 The average miss counts of all tested algorithms in scenario 3

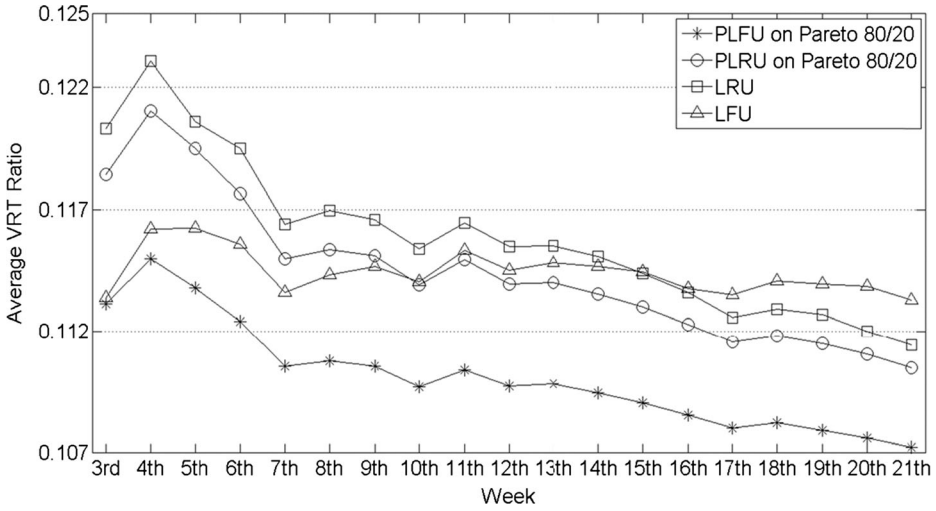


Figure 7 The average VRT ratios of all tested algorithms in scenario 3

4.4 Scenario 4: the entertainment dataset and R=1/3

Figure 8 illustrates the average miss counts of all tested algorithms in scenario 4. The corresponding standard deviations are similar to Table 4. We can see that the trend of all curves in Figure 8 are very similar to those shown in Figure 6, implying that video category did not impact the performance of all the algorithms. Figure 8 also shows that the miss count of PLFU still outperforms those of the other three algorithms. The average miss count of PLFU is about 0.686, 0.639, 0.695 times that of LFU, LRU, PLRU, respectively, during the 21 weeks. Figure 9 shows the average VRT ratios of all algorithms in scenario 4. The corresponding standard deviations are similar to Table 5.

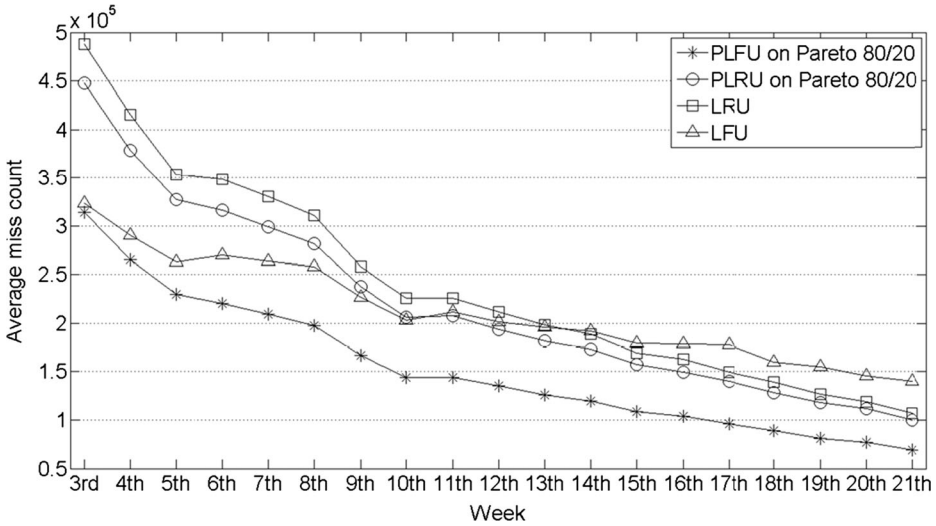


Figure 8 The average miss counts of all tested algorithms in scenario 4

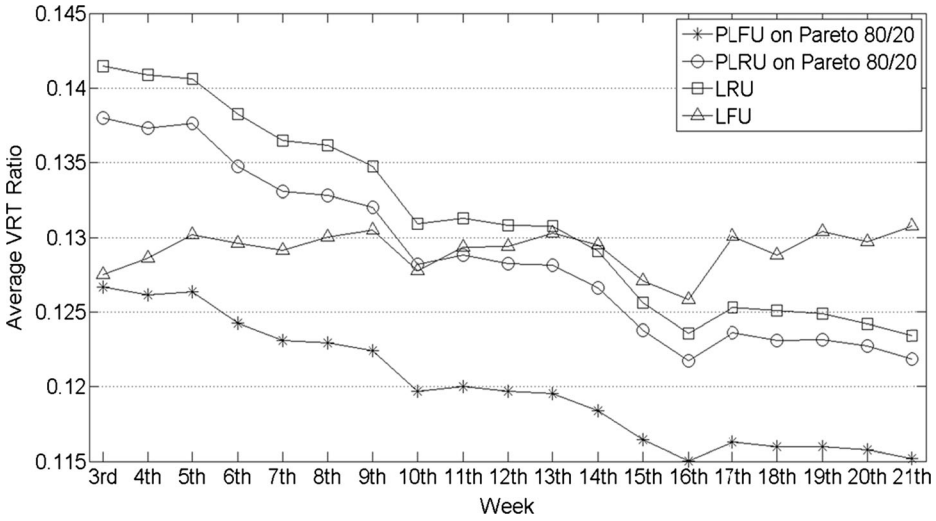


Figure 9 The average VRT ratios of all tested algorithms in scenario 4

By comparing Figure 9 with Figure 7, it is clear that the VRT ratio of LFU was the worst at the end, and it was much higher than those of the other algorithms. Hence, we can conclude that employing LFU is inappropriate for YouTube.

4.5 Time complexity analysis

In this subsection, we analyze the time complexity of each of the four algorithms. As mentioned earlier, before PLFU and PLRU perform, they use the PVD algorithm to sort the access counts of all videos in each video category and then determine popular videos and unpopular videos. In fact, the sorting method used by the PVD algorithm is the Heapsort [22], whose time complexity is $O(n\log n)$ [22], implying that the time complexity of the PVD algorithm is $O(n\log n)$.

Recall that PLFU and PLRU are respectively based on LFU and LRU to make the replace decision, and the time complexities of LFU and LRU are $O(\log n)$ [4] and $O(1)$ [4], respectively. Hence, we can see that the time complexities of PLFU and PLRU are both dominated by the PVD algorithm. Hence, for both PLFU and PLRU, their time complexities are $O(n\log n)$.

Table 6 summarizes the time complexities of the four tested algorithms. Although the time complexities of PLFU and PLRU are slightly worse than those of LFU and LRU, the experimental results shown in sections 4.1 to 4.4 have proved that these

Table 6 The time complexities of all tested algorithms

Algorithm	Time complexity
PLFU	$O(n\log n)$
PLRU	$O(n\log n)$
LRU	$O(1)$
LFU	$O(\log n)$

two algorithms have better performances in terms of miss count and average VRT ratio.

5 Conclusions and Future work

In this paper, we have introduced two Parteo-based cache replacement algorithms, i.e., PLFU and PLRU, for YouTube. Both of them always keep 20% of top popular videos of each video category in Memcached without choosing them as substitute candidates. When Memcached has insufficient space, PLFU evicts the least frequently watched video from Memcached, whereas PLRU evicts the least recently watched video from Memcached. We conducted extensive experiments to evaluate PLFU and PLRU with two known cache replacement algorithms, including LFU and LRU, based on a real-world YouTube trace. The results confirm that employing the Parteo principle enables both PLFU and PLRU to effectively reduce miss count and VRT ratio. Among all tested algorithms, PLFU performs the best in terms of miss count and VRT ratio, regardless of the tested video category and the size of Memcached. In other words, employing PLFU can dramatically reduce the number of video retrievals from the back-end database of YouTube and shorten the video retrieval time. On the other hand, when PLRU is used for a longer time, it provides the second best performance in terms of miss count and VRT ratio among all tested algorithms.

In the future, we would like to study how the four algorithms perform when the arrival time of each video request follows different distribution, such as geometric distribution [11] and uniform distribution [25], and accordingly design an appropriate cache replacement algorithm. These constitute our future work.

Acknowledgments The work was supported by the Ministry of Science and Technology, Taiwan under Grants NSC 101-2628-E-009-024-MY3 and NSC 101-2221-E-009-003-MY3. The authors thank the scholarship of the Sandwich Programme supported by Ministry of Science and Technology, Taiwan and Deutscher Akademischer Austausch Dienst (DAAD) under Grants NSC 102-2911-I-100-524 and NSC 101-2911-I-009-020-2. The authors also want to thank Dr. Jia-Chun Lin for her assistance and suggestions to this paper.

References

1. Abhari, A., Soraya, M.: Workload generation for YouTube. Multimedia Tools and Applications **46**(1), 91–118 (2010)
2. Adhikari V.K., Jain S., Zhang Z.-L.:"YouTube Traffic Dynamics and Its Interplay With a Tier-1 ISP: an ISP Perspective," In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement. ACM, pp. 431–443 (2010)
3. Alexa, http://www.alexa.com/siteinfo/youtube.com. Accessed 23 Nov 2014
4. Algorithms, https://www.usenix.org/legacy/events/usenix01/full_papers/zhou/zhou_html/node3.html. Accessed 23 Nov 2014
5. Breslau, L., Cao, P., Fan L., Phillips G., Shenker S.: "Web caching and Zipf-like distributions: evidence and implications," in: Proceedings of IEEE INFOCOM'99, no.1, New York, USA, pp. 126–134, 21–25 March (1999)
6. Cache Algorithms, http://en.wikipedia.org/wiki/Cache_algorithms#Least_Recently_Used. Accessed 23 Nov 2014
7. Cha, M., Kwak, H. Rodriguez, P. Ahn Y.-Y, Moon, S.: "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. ACM, pp. 1–14 (2007)

8. Cheng, X., Dale C., Liu J.: "Statistics and Social Networking of YouTube Videos," In: Proceeding of the 16th IEEE International Workshop on Quality of Service, pp. 229–238 (2008)
9. Cheng, X., Liu J.: "Load-Balanced Migration of Social Media to Content Clouds," In: Proceedings of the 21st ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp. 51–56 (2011)
10. Daum UCC, http://ucc.daum.net. Accessed 23 Nov 2014
11. Geometric distribution, http://en.wikipedia.org/wiki/Geometric_distribution. Accessed 23 Nov 2014
12. Jose, J., Subramoni, H., Kandalla, K., Wasi-ur-Rahman, M., Wang, H., Narravula, S., Panda, D.K.: "Scalable memcached design for InfiniBand clusters using hybrid transports", In: Proceedings of the 12th IEEE/ACM International Symposium on Cluster. Cloud and Grid Computing **13–16**, 236–243 (2012)
13. Kleinrock, L.: Queueing Systems. In: Theory Vol. 1. Wiley, New York (1975)
14. Labovitz, C., Iekel-Johnson, S., McPherson, D., Oberheide, J., Jahanian, F.: "Internet Inter-domain Traffic," in ACM SIGCOMM Computer Communication Review, vol. 40, no. 4, (2010)
15. Lee, M.-C., Leu, F.-Y., Chen, Y.-p.: PFRF: an adaptive data replication algorithm based on star-topology data grids. Future Generation Computer systems **28**(7), 1045–1057 (2012)
16. Lee, M.-C., Leu F.-Y., Chen, Y.-p.: "Cache Replacement Algorithms for YouTube," in 2014 I.E. 28th International Conference on Advanced Information Networking and Applications (AINA), pp. 743–750 (2014)
17. LiveJournal, http://www.livejournal.com/. Accessed 23 Nov 2014
18. Memcached, http://en.wikipedia.org/wiki/Memcached. Accessed 23 Nov 2014
19. Newman, M.E.J.: Power laws, Pareto distributions and Zipf's law. Contemporary Physics **46**, 323–351 (2005)
20. Page Replacement Algorithm, http://en.wikipedia.org/wiki/Page_replacement_algorithm. Accessed 23 Nov 2014
21. Saab, P.: "Scaling Memcached at Facebook," 12 December 2008. [Online]. Available: https://www.facebook.com/note.php?note_id=39391378919&ref=mf. Accessed 23 Nov 2014
22. Sorting algorithm, http://en.wikipedia.org/wiki/Sorting_algorithm. Accessed 23 Nov 2014
23. Torres, R., Finamore, A., Kim, J., Mellia, M., Munafo, M., Rao, S.: "Dissecting Video Server Selection Strategies in the YouTube CDN," In: Proceeding of the 31st IEEE International Conference on Distributed Computing Systems,. pp. 248–257 (2011)
24. Twitter Engineering, "Memcached SPOF Mystery," Twitter Engineering, 20 April 2010. [Online]. Available: http://engineering.twitter.com/2010/04/memcached-spof-mystery.html. Accessed 23 Nov 2014
25. Uniform distribution (discrete), http://en.wikipedia.org/wiki/Uniform_distribution_(discrete). Accessed 23 Nov 2014
26. User-generated Content, http://en.wikipedia.org/wiki/User-generated_content. Accessed 23 Nov 2014
27. Wiggins, A., Langston J.: "Enhancing the Scalability of Memcached," http://software.intel.com/sites/default/files/m/0/b/6/1/d/45675-memcached_05172012.pdf