http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 12 (21): 6130-6136, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.6130.6136

Cache Replacement Algorithm of Video Sharing System for Mobile Users

Chuanchang Liu and Jia Guo
State Key Lab of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

Abstract: With the advent of powerful mobile devices, the video sharing applications are necessary to supply the needs of mobile users. In this study, we design a video sharing system for mobile users, so that they can share their videos with others. To make this system scalable, we use cache servers to offload the original web server. However, the traditional cache replacement algorithms may not work well in this system due to the size of each video item is different. Thus, we present an improved cache replacement algorithm to adapt to this video sharing system. Experiment results prove the algorithm effective.

Key words: Cache replacement algorithm, video sharing, mobile device

INTRODUCTION

Mobile devices have more features than before. They are no longer used for communication only, they can take photos, record videos, play games and visit the Internet. With the social network becomes more and more popular, sharing feelings with others becomes a part of people everyday life. With the widespread use of photographic equipment, making video clips becomes popular. Mobile users need the video sharing system to share their video clips in the social network.

Hardware of mobile devices, as well as operating systems, development tools and third party libraries, have been evolving rapidly in recent years. But different mobile device platforms have their own ways of data transmission and video media delivery method. We need to find a unified way using the same data interfaces and video media delivery for different mobile device platforms.

In order to improve the speed of user access, cache servers are deployed between the clients and the original HTTP server which are used to offload the original server by serving the clients with video files already pulled from the original content server and stored in the cache server. There are some classic cache replacement algorithms for cache servers but they all have some advantages and disadvantages and may not be suitable for the current system. We design a new cache replacement algorithm based on file size to adapt to this video sharing system.

The remains of the study are organized as follows. Section 2 describes a brief introduction of this video sharing system and the problem of system expansion. After discussing the related works of cache replacement algorithms in section 3, section 4 describes the architecture of this video sharing system and Section 5 describes the improved cache replacement algorithm. Experimental results of different cache replacement algorithms are reported in Section 6. Finally, concluding remarks are given in Section 7.

PROBLEM STATEMENT

The video sharing system contains three servers acting as three different roles. DB server stores meta-data of this system, content server stores all the video files that users upload from their mobile devices and web server provides interfaces for mobile applications to access. The basic physical architecture of this system is shown in Fig. 1.

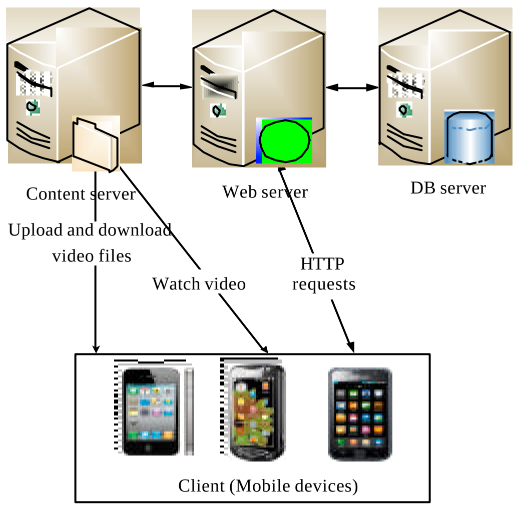

Fig. 1: Architecture of the video sharing system

Corresponding Author: Chuanchang Liu, State Key Lab of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Table 1: Comparison of smart phone platforms

Phone	Apple iPhone	Google android	Windows
HTTP Progressive download	Yes	Yes	Yes
RTSP streaming	No	Yes	Non-native
HTTP Live Streaming	V 3.0+	Not released	No
Windows media HTTP streaming	No	No	Yes
Microsoft silverlight smooth streaming	No	No	Windows Phone 7
RTMP Streaming	No	No	Flash lite

where, PCs can be easily configured to support a broad array of media delivery methods, protocol support varies widely across the major mobile device platforms, as shown in Table 1 (Ma *et al*., 2011).

Flash LiteThere are two very different delivery methods for videos, streaming and progressive download but the end result may look the same to the end user. A standard web server delivers a video file over HTTP (progressive download), while a streaming server opens a conversation with the local machine. In order to maintain the various mobile device platforms, we choose HTTP progressive download as the delivery method for videos (Chattopadhyay *et al*., 2007).

When user watch videos, the system will offer a url of this video on the content server which means the mobile devices will connect directly to the content server. When the users become more and more, the content server will be overloading and the access speed of users will be slow. Another problem is that when the video files become more and more, there will be not enough storage space on just one content server. We introduce web cache server to solve the overloading problem of the original content server.

Web cache servers have many advantages. First, they reduce the waiting time of end users. Moreover, they also reduce the requests for the same content and offload the weight of the whole network. By establishing a buffer path between the cache servers and original web server, it will reduce the load of the original web server. Obviously, it's impossible to store every video files in the cache servers due to the limitation of storage and other problems. When the storage of cache servers is full, we must replace and clean up some content in the cache servers. Some cache replacement algorithms are proposed to solve this problem.

RELATED WORK

There are a number of cache replacement algorithms. We investigate four ones of these: lease recently used (LRU), least frequency used (LFU), frequency based replacement (FBR) and RAND (random replacement).

LRU replaces the item in the cache which has not been used for the longest period of time. This algorithm exploits the principle of temporal locality that items which have been referenced in the recent past will likely be referenced again in the near future (Willick *et al*., 1993).

LFU requires that a reference count be maintained for each item in the cache. When a replacement is necessary, LFU chooses the item which has the lowest reference count. This algorithm uses the history of accesses to predict the probability of a subsequent reference. Items with high counts are more likely to be referenced than items with low counts. Unfortunately, lines with large frequency counts that will not be accessed again tend to remain entrenched in the cache, preventing newer additions to the cache from gathering sufficient reference counts to stay in the cache (Karedla *et al*., 1994).

FBR is a hybrid replacement algorithm, attempting to capture the benefits of both LRU and LFU (Robinson and Devarakonda, 1990). FBR divides the cache into three partitions: A new partition, a middle partition and an old partition. The sizes of these partitions are specified by two parameters F_{new} and F_{old}, F_{new} indicates the percentage of items which are contained in the new section, while F_{old} indicates the percentage of items contained in the old section. When a reference occurs to an item in the new section, its reference count is NOT incremented while references to the middle and old sections do cause the reference counts to be incremented. When an item must be chosen for replacement, FBR chooses it with the lowest reference count but only among those items that are in the old section. This is done because the items in the new and middle sections have not had enough time to build up their reference counts.

RAND chooses among all items in the cache with equal probability. For RAND to work well, it would be required that all items are accessed with equal probability which is not likely to be the case. In effect, RAND provides a kind of lower bound on performance. That is, there is no reason to use any algorithm which performs worse than RAND.

These replacement algorithms specifies which a cache item should be removed when a new item must be entered into an already full cache which means every item has the same size. But in this video sharing system, the size of each video file is different. These cache replacement algorithms may not suitable for the current system. We design a new cache replacement algorithm based on item size for the current system.

ARCHITECTURE OF VIDEO SHARING SYSTEM

Logical architecture: The video sharing platform consists of three layers: Resource Layer, Service Layer and User Layer (Mobile Applications). The logical architecture of this system is shown in Fig. 2.

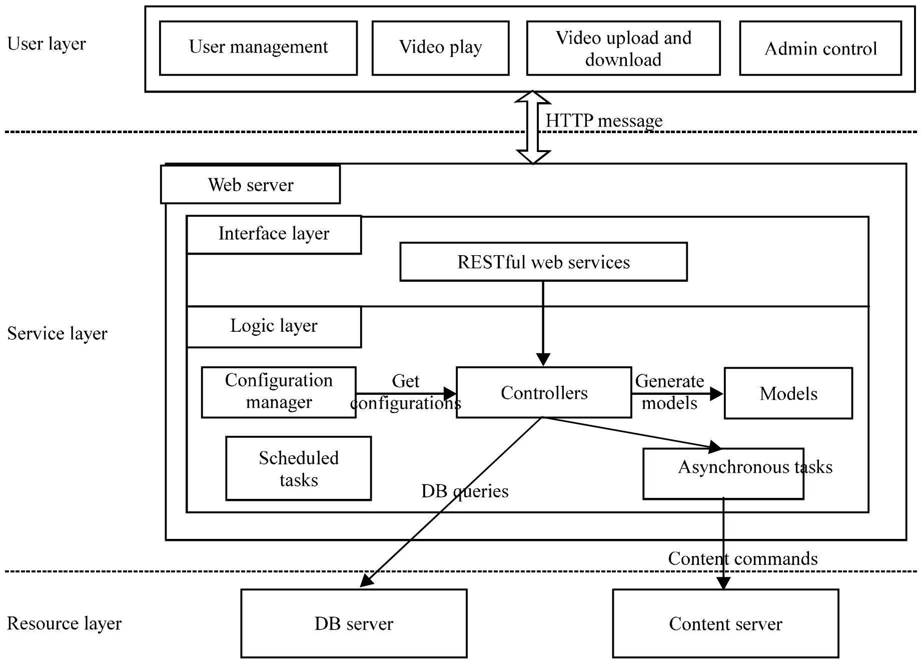

Fig. 2: Logical architecture of video sharing system

Resource layer: Resource layer contains database server and content server. Database server stores meta-data of this system. The meta-data of this system consists of the basic information of ordinary users, the properties of videos, the admin information, the records of video auditing and so on. Content server stores all the video files that users upload from their mobile devices.

Service layer: Service lays in the application deployed in Web Server. The application is separated into two layers: Interface Layer and Logic Layer.

At Interface Layer, RESTful Web Services are used to provide as interfaces to applications on mobile devices. Representational State Transfer (REST) is a style of software architecture for distributed systems such as the World Wide Web. REST has gained widespread acceptance across the Web as a simpler alternative to SOAP and Web Service Description Language (WSDL)-based Web services. Key evidence of this shift in interface design is the adoption of REST by mainstream 2.0 service providers-including Yahoo, Google and Facebook-who have deprecated or passed on SOAP and WSDL-based interfaces in favor of an easier-to-use, resource-oriented model to expose their services.

At Logic Layer, Controllers and Scheduled Tasks are two main components. Controllers are responsible for user requests: list the recommended videos list, view the basic properties of one video, view the comments of one video, set the comment of one video, log in, log out, register, upload videos, download videos, watch videos and so on. Scheduled tasks do things that are independent from user requests. Configuration Manager is used to manage all the data about database and parameters needed in this application. Models are abstractions for data to be presented in mobile applications. They can be serialized in response messages of RESTful Web Services.

User layer: User layer means the mobile applications in different kinds of mobile devices and the admin control web application. Ordinary users could use the application on their mobile devices to manage their information, watch videos, upload and download videos. We aim at three mainstreams mobile operating system: Apple iPhone, Google Android and Windows Phone. Admin control consists of video auditing, ordinary user management, video management, recommended videos setting and system parameter setting.

Physical architecture: We add cache servers to solve the overloading problem. The extended physical architecture of this system is shown in Fig. 3.

The main user scenarios of this system are glance videos' properties, upload and download video files, watch videos. All the user scenarios begin with the HTTP requests from the mobile applications to web server, then the web server queries the meta-data from the DB server and the web server sends back the meta-data to the mobile applications.

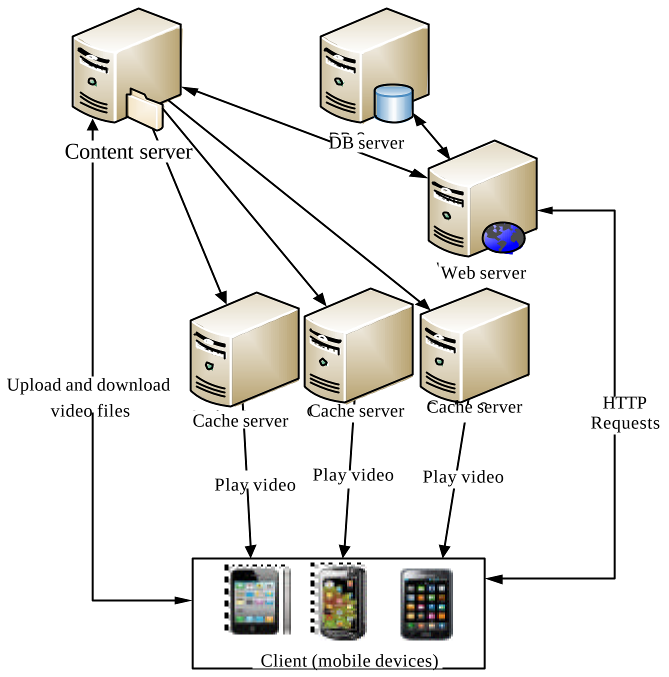

Fig. 3: Physical architecture of video sharing system

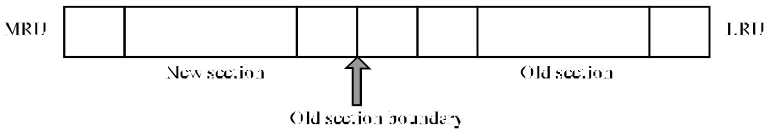

Fig. 4: Two sections of S-LRU cache

In the user upload video scenario, the mobile applications send the HTTP request for the upload path in the content server and then upload the video to the content server. In the user download video scenario, the mobile applications send the HTTP request for the download path in the content server and then download the video from the content server.

In the watch video scenario, the mobile applications send the HTTP request for the video play url in the corresponding cache server. If the corresponding cache server contains this video, it will provide this video to the user. If the cache server doesn't contain this video, this video will be transferred from the content server. When the storage of the cache server is filled, it will remove some video files for the new video file.

CACHE REPLACEMENT ALGORITHM

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar.

The biggest characteristic of the cache server in this video sharing system that is not similar to common cache system is that the size of each item (video file) is different. When the storage of the cache server is full, the cache replacement algorithm will remove more than one video files for the new coming video file. When the new video file is larger, the number of removed video files is also larger, which will reduce the hit rate of the cache server. When the storage size of cache server is not large, the hit rate is the most important metrics to measure the performance of the cache server.

We design a new replacement algorithm called size-based LRU (S-LRU) considering both the benefit of LRU and the item size. S-LRU maintains the LRU ordering of all video items in the cache server but the replacement decision is primarily based upon the item size. To accomplish this, S-LRU divides the cache into two partitions (Fig. 4): A new partition and an old partition. The sizes of these partitions are specified by one

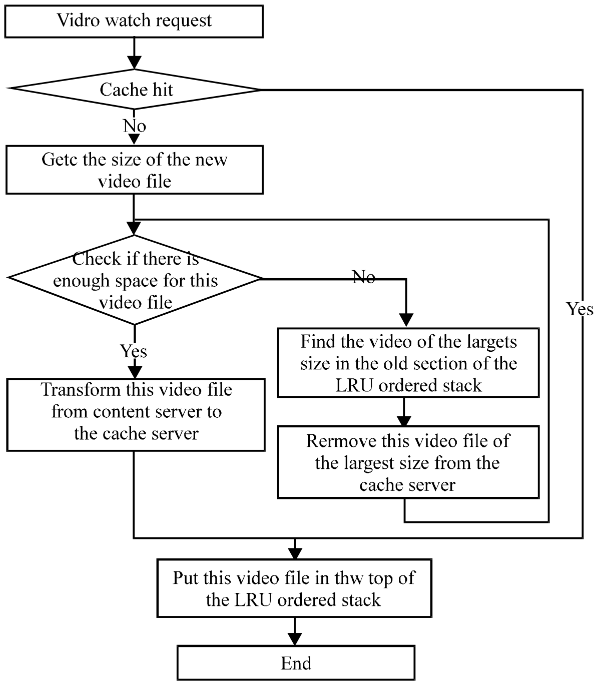
Vidro watch request
Cache hit
No
Getc the size of the new video file
Check if there is enough space for this video file
No
Yes
Yes
Find the video of the largets size in the old section of the LRU ordered stack
Transform this video file from content server to the cache server
Rermove this video file of the largest size from the cache server
Put this video file in thw top of the LRU ordered stack
End

Fig. 5: Flowchart of S-LRU algorithm

parameter to the model. Fold indicates the percentage of items contained in the old section (the Least Recently used (LRU) end). When items must be chosen for replacement, S-LRU chooses the items with the largest size but only among those items that are in the old section of the LRU ordered stack. When Fold = 0, the S-LRU algorithm becomes the LRU algorithm. When Fold = 100, the S-LRU algorithm becomes choosing the item with the largest size to remove.

Figure 5 shows the flowchart of S-LRU algorithm. In the S-LRU algorithm, when cache is not hit and there is not enough space for the new video file, it chooses the largest video file in the old section of LRU ordered stack as the removed video file. After the first remove, if there is still not enough space for the new video file, the S-LRU will select the another video file to remove until the rest space is enough for the new video file. When the cache server size is different, the choice of F_{old} is different. The next section will show the experiment result of hit rate comparison in different value of F_{old}.

EXPERIMENTS

The experiment consists of two parts: the comparison of different value of Fold that affects the hit rate of the S-LRU algorithm and the comparison of the S-LRU and other cache replacement algorithms.

The data of video sizes in the video sharing system are consistent with a uniform distribution. In the experiment, the count of video files in the content server is 1, 000, 000 and the average video size is about 100MB. The total size of all the video files in the content server is about 100TB.

These experiments are all in normal distribution data access. The normal (or Gaussian) distribution is a continuous probability distribution that has a bell-shaped probability density function. The normal distribution is mostly like the real-user access that some videos are very popular with high visits and some videos has very low visits.

Experiment of new LRU in different F_{old}: In this experiment, we change the F_{old} from 5 to 95. The trend graphs are shown in Fig. 6. Each graph has different cache server size from 10TB to 90TB.

From the graphs, we can see that when the cache server size is small (between 10 and 30TB), the larger the number F_{old} is, the higher the hit rate is. At this time, the S-LRU algorithm becomes selecting the largest video files of all to remove. When the cache server size is middle (between 40 and 60TB), the hit rate is high when F_{old} is between 30 and 60. When the cache server size is large (between 70 and 90TB), the hit rate is high when F_{old} is between 10 and 15.

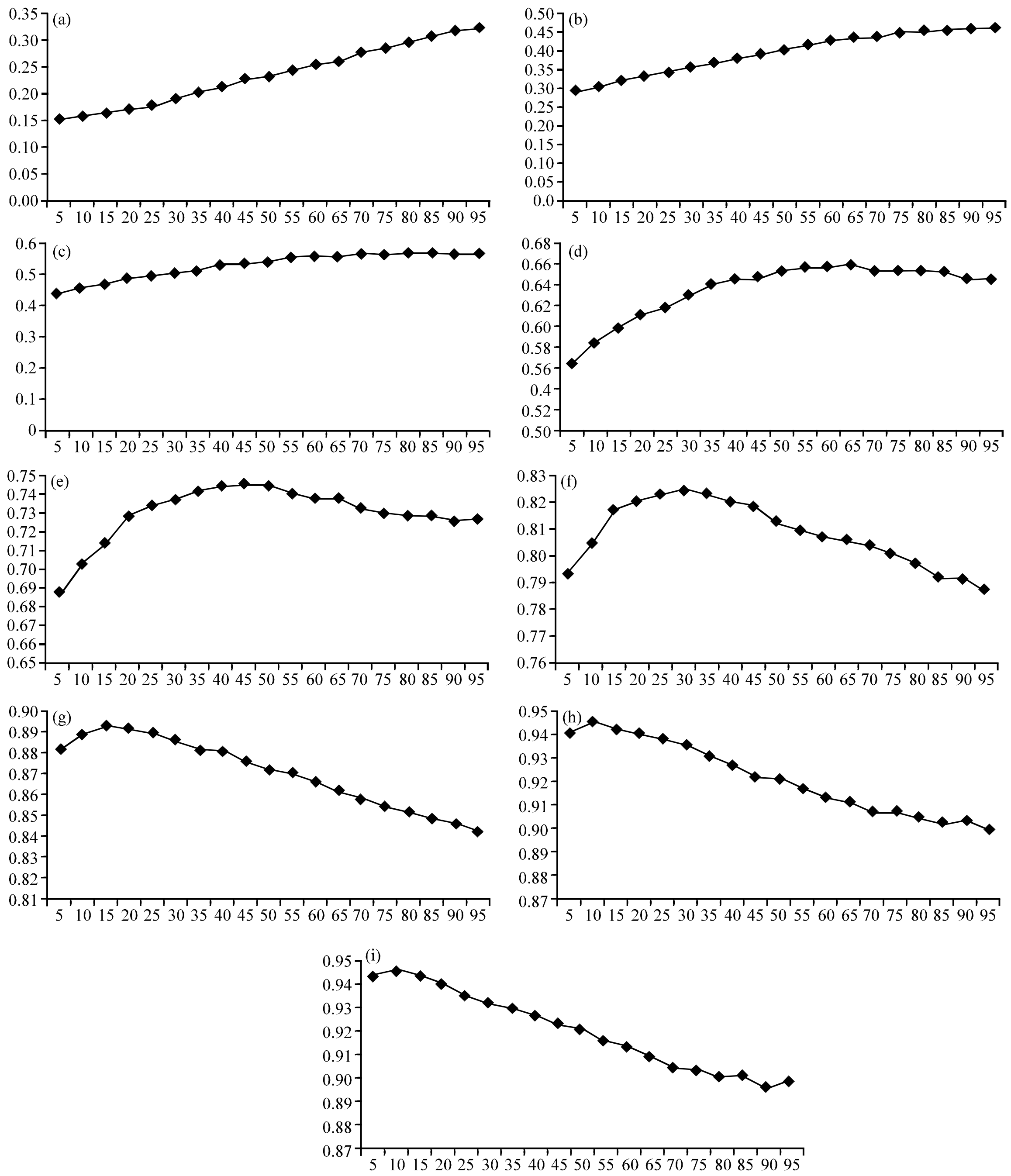

Fig. 6(a-i): Hit rate comparison of different F_{old} (from 5 to 95) in different cache sizes (from 10TB to 90TB), cache size, (a) 10 TB, (b) 20 TB, (c) 30 TB, (d) 40 TB, (e) 50 TB, (f) 60 TB, (g) 70 TB, (h) 80 TB and (i) 90 TB

On balance, we choose F_{old} = 60 of the S-LRU algorithm in the next experiment to compare with other cache replacement algorithms.

Comparison experiments of five cache replacement algorithms.

Experiments simulate five cache replacement algorithms: S-LRU, LRU, LFU, FBR and RAND. S-LRU sets F_{old} = 60, meaning that the old section contains the bottom 60 percent of the LRU stack. FBR sets F_{new} = 25 meaning that old section contains the top 25% of the

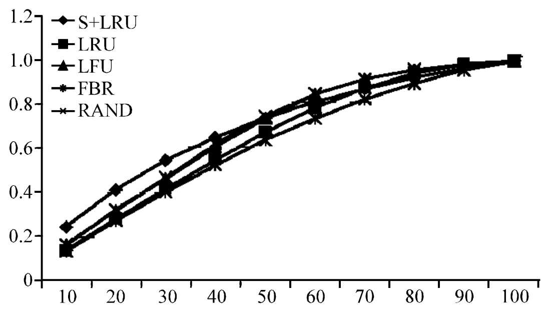

Fig. 7: Hit rate comparison of cache replacement algorithms

LRU stack, while F_{old} = 60 meaning that old section contains the bottom 60 percent of the LRU stack. These parameter choices of FBR are consistent with where they were obtained through experimentation. The hit rate comparison of S-LRU with other algorithms is show in Fig. 7.

CONCLUSIONS

With the fast development of Mobile Internet, mobile devices provide people new way to share their videos ubiquitously. The study presents an video sharing system for users to access by different kinds of mobile devices. We also design a new size-based cache replacement algorithm (S-LRU) for this system to improve the performance of the cache server. Experiments show that the S-LRU cache replacement algorithm is suitable for the current system.

ACKNOWLEDGMENT

This study is supported by the National Grand Fundamental Research 973 Program of China under Grant No.2011CB302506 and the Technology Development and Expriment of Innovative Network Architecture under Grant No.CNGI-12-03-007.

REFERENCES

Chattopadhyay, S., L. Ramaswamy and S.M. Bhandarkar, 2007. A framework for encoding and caching of video for quality adaptive progressive download. Proceedings of the 15th International Conference on Multimedia, September 24-29, 2007, Augsburg, Germany, pp: 775-778.

Karedla, R., J.S. Love and B.G. Wherry, 1994. Caching strategies to improve disk system performance. Computer, 27: 38-46.

Ma, K.J., R. Bartos, S. Bhatia and R. Nair, 2011. Mobile video delivery with HTTP. IEEE Commun. Mag., 49: 166-175.

Robinson, J.T. and M.V. Devarakonda, 1990. Data cache management using frequency-based replacement. Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, May 22-25, 1990, USA., pp: 134-142.

Willick, D.L., D.L. Eager and R.B. Bunt, 1993. Disk cache replacement policies for network fileservers. Proceedings of the 13th International Conference on Distributed Computing Systems, May 25-28, 1993, Pittsburgh, PA., USA., pp: 2-11.