International Journal of Innovative
Computing, Information and Control
Volume 11, Number 2, April 2015

ICIC International ©2015 ISSN 1349-4198
pp. 555–568

COOPERATIVE WEB PROXY CACHING FOR MEDIA OBJECTS BASED ON PEER-TO-PEER SYSTEMS

Waheed Yasin, Hamidah Ibrahim, Nur Izura Udzir
and Nor Asilah Wati Abdul Hamid

Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
43400 Serdang, Selangor D.E., Malaysia
waheedos80@yahoo.com; { hamidah.ibrahim; izura; asila }@upm.edu.my

Received April 2014; revised August 2014

Abstract. *Web proxy caches are used to improve the performance of the World Wide Web (WWW). Many advantages can be gathered from caching such as improving the hit rates, reducing network traffic, and alleviating loads on origin servers. On the other hand, retrieving the same object many times consumes the network bandwidth. Thus, in order to overcome this limitation, in this work, a cooperative web caching approach for media objects based on peer-to-peer systems is proposed. Tow performance metrics are used that are Hit Ratio (HR) and Byte Hit Ratio (BHR). A simulation is carried out to study the affects of cooperative caching on the performance of web proxy caching policies. The results show that cooperative caching improves the performance of web proxy caching policies in delivering media objects.*

Keywords: Web proxy, Simulation, Caching policies

1. Introduction.

Web proxy caching is the process of storing web objects close to the end-users. Thus, it reduces web objects delivering latency to the end-users. Furthermore, it utilizes the network bandwidth which is a key point for network administrators. Also, it contributes in alleviating loads on origin servers [1].

Web caching might be performed in three different stages that are browser stage, proxy stage, and web server stage [2]. In our previous work [3], we have presented the affects of caching in browser stage on the performance of web objects delivery.

Retrieving the same media objects from the origin servers consumes the network bandwidth. Thus, an approach based on cooperative web caching is needed in order to utilize the network bandwidth and to alleviate the load on origin servers.

The aim of this work is to present a cooperative web caching policy for media objects based on peer-to-peer systems.

In this work, we will take the advantages of both systems client-server and peer-to-peer systems where peers' caches contents are shared in order to enhance the performance of web caching policies. The proposed approach is evaluated by trace-driven simulation and compared with the most relevant web proxy caching policies.

This paper is organized as follows. Section 2 presents an overview of web caching. The proposed approach is presented in Section 3. The experiment is presented in Section 4. The results and discussion are presented in Section 5. Section 6 presents recent related works. Section 7 presents the conclusions and the future works.

2. Web Caching.

The process of storing data in an intermediate media is called web caching and it is used for responding to future queries rather than fetching data from origin sources.

A cache hit indicates that the system responds to the query from the cache; while a cache miss indicates that the cache does not have the required data. Many benefits can be achieved from applying caching such as reducing the load of the system and the latency. Moreover, it utilizes the network bandwidth.

In this section, the benefits and limitations of web caching are presented. Furthermore, cache replacement factors are presented in addition to some web caching policies.

2.1. **Web caching benefits.** Many benefits can be gathered from applying web caching system in the network. The main benefits of web caching are listed below:

- Web caching allows responding to users' queries from the cache storage which results in reducing the latency. That is because the response time which is consumed to deliver the web objects from the cache is less than the consumed time for delivering them from the origin server.
- Web caching reduces the system load because there is no need to generate new information. A consistent copy of this information can be retrieved from the cache.
- Retrieving same web objects from the origin server many times wastes the outgoing link (uplink). Thus, web caching is used to save the consumption of the network bandwidth, because there is no need for retrieving web objects many times.

2.2. **Web caching limitations.** Although there are many benefits of web caching, there are some limitations that have to be considered when caching is used in the system. Some of these limitations are listed below:

- Data Privacy should be considered in web caching because there is some data considered as sensitive data. These sensitive data should be manipulated in a different way. An approach that might be used is categorizing data into two groups that are cacheable data and non-cacheable data. The cache only keeps the cacheable data.
- Data Consistency in the cache is a key point that should be considered. Web objects might be updated in origin servers that results in invalidity of the cached data. Thus, the cache is responsible to deliver this updated data. In [5], more details are in the area of cache consistency.
- A cache has a limited size which becomes full with web objects. In this case, a replacement decision has to be taken when a new objet has to be cached. A replacement mechanism is defined as the process of removing old web objects according to a certain criteria, and replaces them with new arrivals. Many factors should be considered when a replacement decision has to be taken [7].

2.3. **Caching factors.** A web caching policy uses a replacement mechanism which refers to the process that occurs when the cache becomes full and there is no enough space for new items, which leads to the task of removing old items to make spaces for new ones. It is common that a web caching algorithm is defined according to the replacement mechanism that is used by the cache. Caching replacement factors are listed below:

- Frequency factor: total number of requests of a web object.
- Recency factor: time since the last use of a web object.
- Cost of fetching a web object: cost to fetch a web object from the origin source including bandwidth, processing, and other resources.
- Size factor: size of the web object.
- Modification time factor: time since the last modification.
- Expiration time factor: time as soon as the web object becomes useless and it is able to be replaced.

2.4. **Web caching policies.** Web caching policies are categorized into five categories [6] that are listed below:

- Recency-based web caching policies.
- Frequency-based web caching policies.
- Randomized caching policies.
- Function-based web caching policies.
- Size-based web caching policies.

In this work, some of the web caching policies are simulated that are listed below:

- Least Recently Used (LRU). The basic approach of LRU is to first remove the web objects that are least recently used from the storage of the cache. Thus, LRU has to track these web objects by implementing a recency index for each object, which indicates the time since the last use of the item. The least used one will be replaced with a new coming object. LRU is considered as one of the common mechanisms that are used in data replacement. Many variants of LRU have been proposed in the literature such as SVM-LRU [7] and NBLRU [8].
- Least Frequently Used (LFU). The basic approach of LFU is to first remove the web objects that are least frequently used from the cache. Thus, LFU has to track these web items by implementing a frequency index for each object, which indicates the total number of requests. A web object with the least index value will be replaced with a new coming item. Many variants of LFU have been proposed in the literature such as LFU-DA [5] and LFU-Aging [7].
- Greedy Dual Size (GDS). It has been proposed to consider the differences of objects' sizes that have different costs to fetch them from origin sources. GDS uses an index which is defined as the cost of fetching the object to the size of that object in bytes. The object with the minimum index value is replaced [9].

2.5. **Media objects characteristics.** Media objects have many characteristics that should be considered. For example, full caching of media objects may lead to consume the storage of the cache in keeping few media objects because of its limited capacity. In [6], media objects characteristics are listed. Some of them are listed below:

- Media object size. Media objects have a huge data volume. For example, a media object might have a size of more than 100 MB, while a text object might have only 100 KB.
- Bandwidth consumption. Media objects delivery consumes a large part of the network bandwidth because of their huge sizes.
- Write-Once-Read-Many (WORM) principle. Media objects usually follow the WOR M principle. Thus, cache consistency is not considered as a main issue.
- Users' interactions. During media object delivery, users might want to do some interactions such as rewind, pause, and fast forward.
- Long playback durations. Long play durations are considered as an advantageous feature of media objects because it can be played before completing transferring the file.
- Timeliness constraints. Time characterizes media objects and plays a key role in the Quality of Service (QoS).

3. **Proposed Approach.**

The aim of this work is to present a cooperative web caching policy for media objects based on peer-to-peer systems. The proposed approach takes the advantages of both systems client-server and peer-to-peer systems where peers' caches contents are shared in order to enhance the performance of web caching policies. A flowchart of the proposed approach is illustrated in Figure 1.

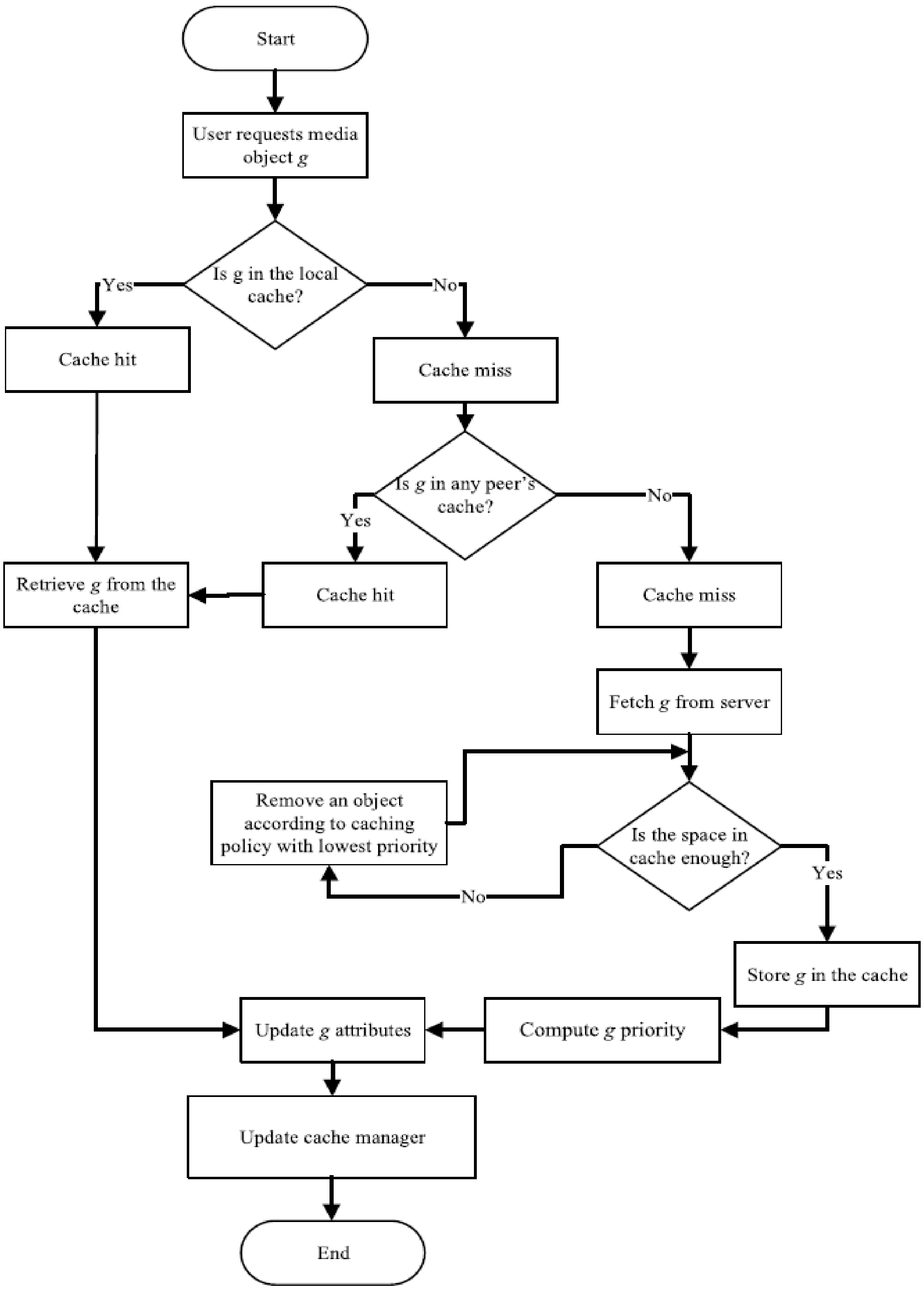

FIGURE 1. Proposed flowchart

Assume that a user requests a media object g. If g exists in the local cache or any other peers cache, the case is a cache hit, and g is retrieved from the cache. The attributes of g are updated for computing the priority of g. Thus, object g is reordered in the cache list based on its new priority. In the case of a cache miss, object g is fetched from the origin server. Then, the attributes of g are collected to compute its priority. The priority of g is computed depending on the web caching policy before it is stored in the cache.

3.1. **Peer-to-peer overlay constructing.** There are two different methods for configuring a peer-to-peer system that are unstructured peer-to-peer systems and structured peer-to-peer systems. In an unstructured peer-to-peer system, a pre-existing infrastructure is not required; also, nodes are dynamically connected to and disconnected from the system such as Mobile Ad Hoc Network (MANET). In structured peer-to-peer system, a pre-existing infrastructure is required such as having access points. In this work, a structured peer-to-peer system is used. Peers in structured peer-to-peer systems maintain information about resources offered by neighbors. Thus, fewer messages are needed for answering queries. Moreover, structured peer-to-peer systems provide both low latency and load balancing. These systems are based on Distributed Hash Table (DHT) in which hash functions are used to map peers and shared content references. Chord [17] is a protocol which is adopted for the structured peer-to-peer system. Chord is based on DHT which uses one dimensional m-bit hash function and organizes peers using ring topology. Each peer has a Chord Identifier which is ranged from 0 to $2^m - 1$. Moreover, each peer has a successor and a predecessor. The successor to a peer is the next peer in the identifier ring in a clockwise direction, while the predecessor is in the opposite direction. For example, the successor of peer 1 is peer 2, while the predecessor of peer 1 is peer 0. On the other hand, peers might connect or disconnect the network freely which leads that a peer records a whole segment of the ring adjacent to it in order to answer queries. Then messages can be routed with a simple routing protocol. An example of Chord protocol for structured peer-to-peer systems is illustrated in Figure 2.

3.2. **Peers' contents sharing.** In peer-to-peer approach, a peer may receive a web object from the origin servers, proxy caches or other peers. Furthermore, peers may form a kind of multicast trees. In cooperative caching, peers share their caches' contents in order to respond to users' queries. For example, assume a media file which is called

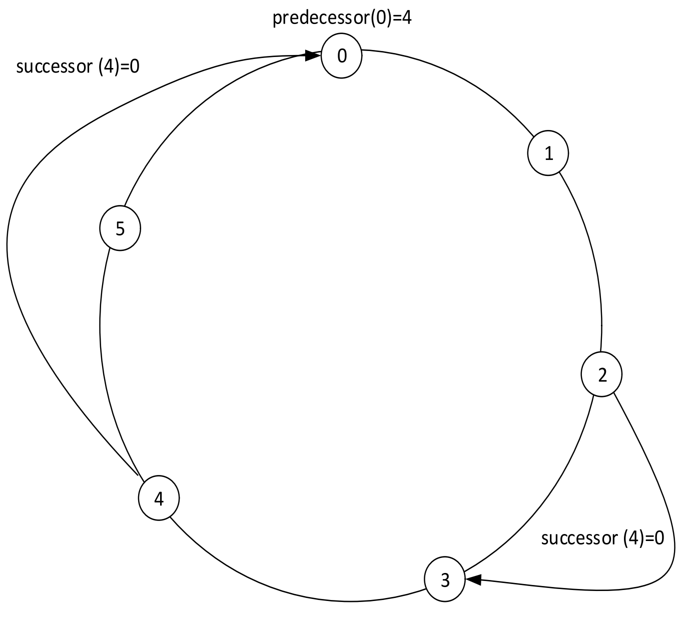

FIGURE 2. An example of Chord protocol for structured peer-to-peer systems

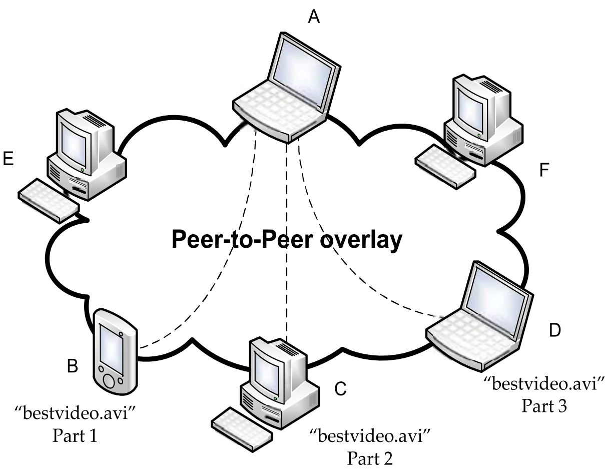

FIGURE 3. An example of file sharing in peer-to-peer system

"bestvideo.avi" is divided into three parts that are separated on three peers B, C, and D as illustrated in Figure 3. When this file is requested by peer A, the file is collected from the peers that have the file parts. This is because of many reasons such as the time which is consumed to download the file from the peers is less than the time which is consumed the file from the origin server. Also, the uplink is kept to be used by other requests. Thus, peers are working as lightweight proxies [15].

3.3. **Cooperative web caching.** In this step, the web proxy caching proxy caching is selected to be integrated with other peers' caches. That means a web proxy caching policy considers the contents of peers' caches before it takes a cache replacement decision. The procedure of of selecting web caching policy integrated with peers' contents is called cooperative web caching. The signaling overhead that is generated between peers is ignored because it does not have a big problem in terms of performance [16]. In other words, some Internet Service Providers (ISPs) take into account the amount of download and upload traffic that are passing through the uplink of the gateway when they issue users' internet bills; while the internal traffic is not considered. As we mentioned earlier, In this work, three web proxy caching policies are simulated that are LRU, LFU, and GDS.

4. **Experiment.** The following tools are used:

- Microsoft Visual Studio 2010 Express (Version 10.0.30319.1 RTMRel).
- Microsoft .NET Framework (Version 4.0.30319 RTMRel).
- Microsoft Office Access 2007 (Version 12.0. 4518.1014 MSO 12.0.4518.1014).
- WinZip version 16 Pro(9691)-32-bit.
- Notepad (Version 6.1).
- GNUPLOT (Version 4.6).

4.1. **Raw data collection.** First, data for the proxy logs and traces for the requested web objects are gathered from several proxy servers of the IRCache network. More details about IRCache network are available on `http://www.ircache.net`. Each entry of a proxy log file contains ten fields that are listed below:

- Timestamp. The time when the client's socket is closed. The format is "Unix time" with millisecond resolution.
- Elapsed Time. The elapsed time of the request, in milliseconds. This is time between the accept() and close() of the client socket. For persistent HTTP connections, this is the time between reading the first byte of the request, and writing the last byte of the reply.
- Client Address. A random IP address identifying the client. The client-to-address mapping stays the same for all requests in a single log file. The mapping is not the same between log files.
- Log Tag and HTTP Code. The Log Tag describes how the request was treated locally (hit, miss, etc.). All the tags are described in the Squid FAQ. The HTTP status code is the reply code taken from the first line of the HTTP reply header. Non-HTTP requests may have zero reply codes.
- Size. The number of bytes written to the client.
- URL. The requested URL. CGI query arguments (anything following a '?') are not logged.
- The requested URL.
- User ID. Always '–' for the IRCache logs.
- Hierarchy Data and Hostname. A description of how and where the requested and Hostname object was fetched.
- Content Type. The Content-type field from the HTTP reply.

Table 1 shows an example of the IRCache proxy logs file.

Table 1. An example of IRCache proxy logs

Timestamp	Elapsed Time (ms)	Client Address	Log Tag and HTTP Code	Size (byte)	Request Method	URL	User ID	Hierarchy Data and Hostname	ContentType
1362731475.948	207	118.205.12.90	TCP_CLIENT_HIT/200	2180	GET	http://cdn1b.pics.hardsextube.com/2009/01/12/121700.1.160.120.jpg	–	DIRECT/200.147.68.8	application/vnd.apple.mpegurl
1362731475.992	189	118.205.12.90	TCP_CLIENT_REFRESH_MISS/200	397	GET	http://cdn1b.pics.hardsextube.com/2010/03/14/310695.1.160.120.jpg	–	DIRECT/200.221.7.95	application/vnd.apple.mpegurl
1362750287.096	248	246.152.90.84	TCP_MISS/200	37121	GET	http://p2.xhamster.com/000/019/798/411_160.jpg	–	NONE/-	image/gif
1362762251.815	101	118.205.12.90	TCP_CLIENT_REFRESH_MISS/200	361	GET	http://p2.xhamster.com/000/020/111/353_100.jpg	–	DIRECT/199.117.103.83	image/jpeg

4.2. **Data pre-processing.** The second step is called pre-processing where the inappropriate and invalid logs such as un-cacheable requests and entries with unsuccessful HTTP status codes which are removed from the proxy log files. Also, the unnecessary fields are ignored because they do not have any effect on caching policy such as Log Tag, HTTP Code, Request Method, User ID, Hierarchy Data and Hostname. On the other hand, in order to reduce the simulation time, each URL is replaced with an integer identifier that

```
While ( !eof(LogFile) )
Begin
        Request=Read(LogFile); // read LogFile line by line
        If( Request.status.endswith("200") &&
            Request.request_method.equals("Get") )
            Begin
                    If( ! Request.url.contains("?") &&
                        ! Request.url.contains ("cgi-bin"))
                        Begin
                                url_id =Convert2integer(Request. url);
                                Write (processed_LogFile, Request.url_id,
                                          Request.timestamp, Request.elapsed_time,
                                          Request.size, Request.Client _Address,
                                          Request.type)
                        End
            End
End
```

FIGURE 4. A pseudo-code for the data pre-processing

TABLE 2. An example of log entries after data pre-processing

Timestamp	Elapsed Time (ms)	Client Address	URL ID	Size (Byte)
1362705584.099	115	10	1	426
1362705584.915	201	10	2	4574
1362705584.998	120	10	3	7329
1362705585.16	116	10	4	3960
1362705585.16	8777	11	6	2614
1362705585.208	110	11	5	3600
1362705585.247	116	11	4	3960
1362706447.423	8777	10	6	2614
1362706447.438	115	11	1	426
1362706448.057	120	11	3	4574

is called "URL ID". Also, the Client Address is replaced with an integer identifier. Figure 4 illustrates a pseudo-code of the pre-processing step; while Table 2 lists an example of proxy logs after the pre-processing step.

4.3. **Simulation.** In [14], we developed a trace-driven simulation tool that is called Windows Web Proxy Caching Simulation (WWPCS) that is build using Microsoft Visual C++ 2010 Express for evaluating web caching policies that are LRU, LFU, and GDS. WWPCS uses the revised proxy log file as an input and generates files that contain the HR and BHR as outputs. The log file was collected from BO2 proxy server on the 18th March 2013. First, the raw data is imported by a database management system. Then, the raw data is pre-processed as mentioned earlier. After pre-processing the generated trace file is exported to a text file.

WWPCS uses the generated trace text file as an input. On the other hand, in order to run the simulation, there are some parameters have to be set before the simulation run that are the maximum cache size (Infinite Cache) in GB and a warm up (Warmup) parameter in MB. Also, the web cache policy has to be determined. WWPCS starts with 1 MB and increases according to a base 2 logarithmic scale.

In this work, WWPCS is improved to take the advantages of cooperative caching. It generates output files which are used to calculate the HR and the BHR for the selected web caching policy before and after working cooperatively.

4.4. **Performance evaluation.** The performance of web caching policy might be measured using many metrics. In this simulation, two main performance metrics are used which are the HR and the BHR because they are the most widely used metrics for evaluating the performance of web proxy caching policies [8]. HR is defined as a percentage of the total number of requests satisfied by the cache divided by the total number of requests; while, BHR is the total number of bytes found in the cache divided by the total number of bytes requested by the user within an observation period. BHR measures how much bandwidth the cache has saved. Let N be the total number of requests of web objects and $h_i = 1$ if the request i is in the cache, while $h_i = 0$ otherwise. Mathematically, HR is defined as Equation (1), while BHR is defined as Equation (2), where S_i is the size of the i^{th} request.

$$HR = \frac{\sum_{i=1}^{N} h_i}{N} \tag{1}$$

$$BHR = \frac{\sum_{i=1}^{N} S_i h_i}{\sum_{i=1}^{N} S_i} \tag{2}$$

5. **Results and Discussion.** In this section, the results that are gathered from running the simulation are presented. For simplicity the word Cooperative is added to the web caching policy in order to refer to the proposed approach.

Figure 5 shows the performance of Cooperative-LRU in terms of HR. It is compared to the performance of normal LRU, where the infinite cache size is 32 GB; while Figure 6 shows the performance of Cooperative-LRU in terms of BHR. Also, it is compared to the performance of normal LRU, where the infinite cache size is 32 GB.

Figure 7 shows the performance of Cooperative-LFU in terms of HR. It is compared to the performance of normal LFU, where the infinite cache size is 32 GB; while Figure 8 shows the performance of Cooperative-LFU in terms of BHR. Also, it is compared to the performance of normal LFU, where the infinite cache size is 32 GB.

Figure 9 shows the performance of Cooperative-GDS in terms of HR. It is compared to the performance of normal GDS, where the infinite cache size is 32 GB; while Figure 10 shows the performance of Cooperative-GDS in terms of BHR. Also, it is compared to the performance of normal GDS, where the infinite cache size is 32 GB.

Referring to Figure 5, it can be observed that Cooperative-LRU performs better than LRU in terms of HR. For example, the HR of Cooperative-LRU is around 28.5% when the cache size is 1 MB; while the HR of LRU is around 28.1% for the same cache size.

Referring to Figure 6, it can be observed that Cooperative-LRU performs better than LRU in terms of BHR. For example, the BHR of Cooperative-LRU is around 29.19% when the cache size is 64 MB; while the BHR of LRU is around 28.56% for the same cache size.

Referring to Figure 7, it can be observed that Cooperative-LFU performs better than LFU in terms of HR. For example, the HR of Cooperative-LFU is around 28.5% when the cache size is 1 MB; while the HR of LFU is around 28.2% for the same cache size.

Referring to Figure 8, it can be observed that Cooperative-LFU performs better than LFU in terms of BHR. For example, the BHR of Cooperative-LFU is around 29.24% when the cache size is 64 MB; while the BHR of LFU is around 28.60% for the same cache size.

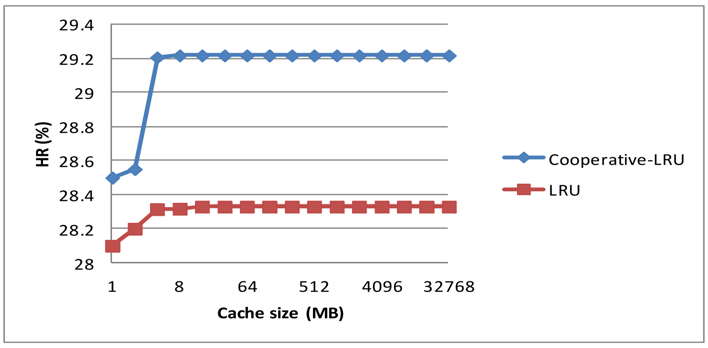

Figure 5. A comparison between the HR of Cooperative-LRU and the HR of LRU, where Infinite Cache = 32 GB and Warmup = 100 MB

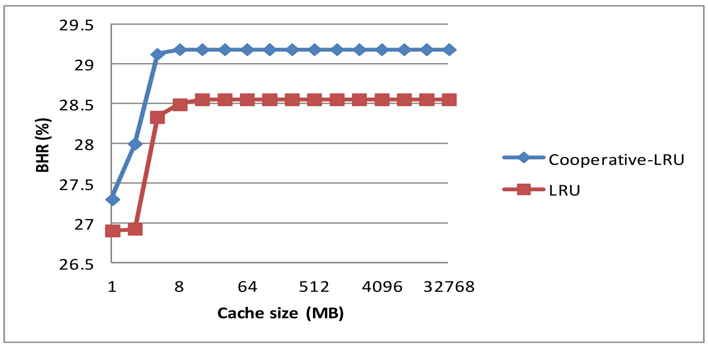

Figure 6. A comparison between the BHR of Cooperative-LRU and the BHR of LRU, where Infinite Cache = 32 GB and Warmup = 100 MB

Referring to Figure 9, it can be observed that Cooperative-GDS performs better than GDS in terms of HR. For example, the HR of Cooperative-GDS is around 30% when the cache size is 1 MB; while the HR of GDS is around 29.76% for the same cache size.

Referring to Figure 10, it can be observed that Cooperative-GDS performs better than GDS in terms of BHR. For example, the BHR of Cooperative-GDS is around 29.27% when the cache size is 64 MB; while the BHR of GDS is around 29.06% for the same cache size.

Also, it can be observed that Cooperative-GDS achieved slightly higher HR and BHR compared to Cooperative-LFU and Cooperative-LRU because Cooperative-GDS takes into account more than one factor when a cache replacement decision is required.

In computer science, a strategy that produces a high HR can maximize satisfaction of users requests from proxy cache, and minimize average request latency. However, a strategy with a higher BHR is used if minimizing of the outside network traffic is more desirable.

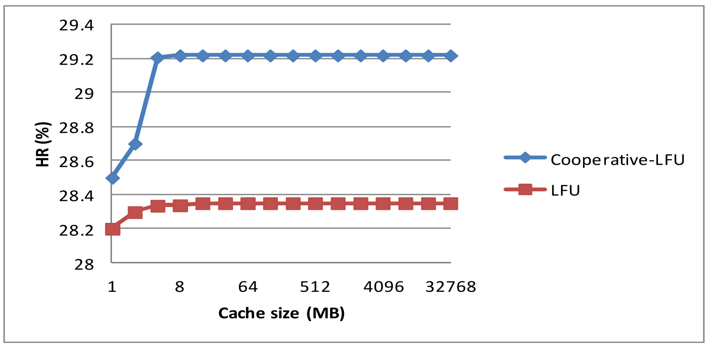

FIGURE 7. A comparison between the HR of Cooperative-LFU and the HR of LFU, where Infinite Cache = 32 GB and Warmup = 100 MB

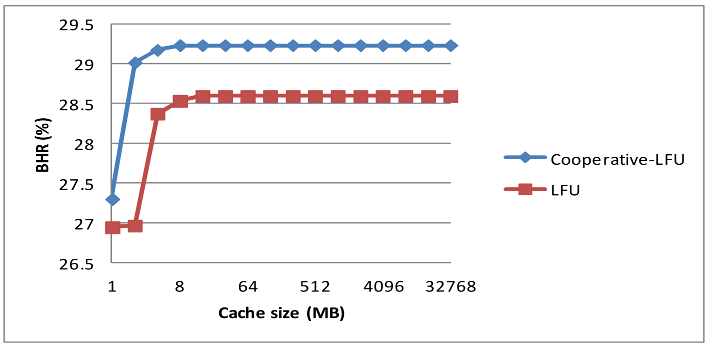

FIGURE 8. A comparison between the BHR of Cooperative-LFU and the BHR of LFU, where Infinite Cache = 32 GB and Warmup = 100 MB

Referring to the results, it can be observed that when the cache size increased, the HR and BHR are increased for all algorithms. On the other hand, the increasing percentage was reduced when the cache size increased. In a certain cache size, the performance of both HR and BHR became stable. Also, when the cache size was small, objects replacement were frequently required. Thus, the affects of the proposed approach on the performance of replacement policy were clearly noted.

In summary, the experimental results show that the proposed approaches Cooperative-LRU, Cooperative-LFU, and Cooperative-GDS significantly improved the performance of the conventional web caching policies.

6. **Related Works.**

Many researches have been conducted in the area of web proxy caching. In this section, recent related works are reviewed.

In [6], an overview of media streams caching in peer-to-peer systems has been presented.

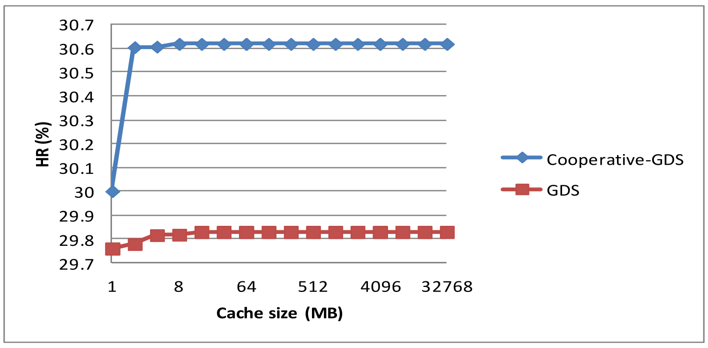

FIGURE 9. A comparison between the HR of Cooperative-GDS and the HR of GDS, where Infinite Cache = 32 GB and Warmup = 100 MB

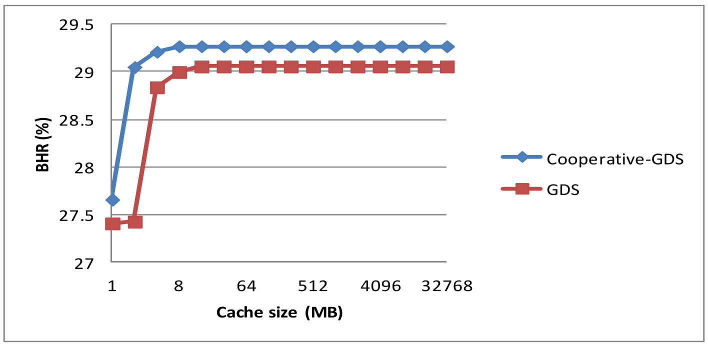

FIGURE 10. A comparison between the BHR of Cooperative-GDS and the BHR of GDS, where Infinite Cache = 32 GB and Warmup = 100 MB

In [3], the affects of web caching on the delivery of web objects in the browser stage have been presented. Three web browsers have been used in the experiment that are: Windows Internet Explorer version 9.0.8112.16421, Mozilla Firefox version 14.0.1, and Google Chrome version 21.0.1180.83 m. The results show that web caching in browser stage has positive effects on the network by reducing bandwidth requirements and improves the delivery of web objects.

In [4], a synthetic workload generator called ProWGen has been presented for simulation evaluation of web proxy caches. ProWGen tool considers three characteristics of web workload that are document popularity distribution, document temporal locality, and the correlation between the size and popularity of the document. Furthermore, ProWGen tool simulates three web caching polices that are LRU, LFU-A, and GD-Size; however, it is a Unix-based tool.

In [11], a trace-driven simulator for evaluating the performance of web proxy caching policies has been presented. The proposed tool has studied the effects of workload characteristics such as object size, recency, and frequency on the performance of web proxy caches and their replacement policies. On the other hand, five months have been spent for collecting proxy logs that are around 117 million logs.

In [12], a windows-based simulator for caching and prefetching has been presented. The proposed tool takes Squid traces as inputs and simulates the functionality of different web caching policies. The outputs of the simulator are saved in text files. The proposed tool has been built using the Borland C++ Builder 2006 IDE; however, WWPCS [14] is built using Microsoft Visual Studio 2010 Express which makes it more compatible for windows operating systems since they are Microsoft productions.

In [13], a packet-level simulation for studying the optimal web proxy cache placement has been performed based on NS-2 network simulations. The study has been carried out considering the network-level effects on the user-level web performance.

In [15], a systematic review of file sharing in peer-to-peer systems has been presented.

In [16], a cooperative caching approach for broadcast-based systems based on peer-to-peer system has been presented. The proposed approach has assumed that clients do not send any query and rather they wait for the sever's broadcast.

7. Conclusions and Future Work.

Web proxy caching improves the performance of WWW by storing web objects close to end-users. Also, it reduces the load on the origin servers. Moreover, it saves the network bandwidth. In this work, a new approach based on cooperative caching has been presented. The clients in this approach construct a peer-to-peer system for exchanging files. Structured peer-to-peer system that is based on Chord protocol has been adopted.

On other hand, simulations play key roles in studying and analyzing the behavior of computer networks under different conditions. Thus, in this work, a simulation has been carried out in order to study and analyze the affects of the proposed approach on the performance of traditional web proxy caching policies that are LFU, LRU, and GDS.

The results show that both of the BHR and HR of the proposed approach have better performance than normal web caching policies specially for small cache sizes.

The proposed approach applies Chord as a peer-to-peer protocol which uses a simple routing system that does not scale. This drawback might result in using additional routing information which is called fingers that are stored at every single peer. On the other hand, we assumes that all proxy caches have the same cache size.

Enabling supervised machine learning techniques to introduce intelligent cooperative web caching approach might be considered as future work.

Acknowledgments. The traces and proxy log files have been provided by the National Science Foundation (grants NCR-9616602 and NCR-9521745), and the National Laboratory for Applied Network Research (NLANR).

REFERENCES

[1] P. R. Jelenkovic and A. Radovanovic, Asymptotic optimality of the static frequency caching in the presence of correlated requests, *Operations Research Letters*, vol.37, no.5, pp.307-311, 2009.

[2] C. Kumar and J. B. Norris, A new approach for a proxy-level web caching mechanism, *Decision Support Systems*, vol.46, no.1, pp.52-60, 2008.

[3] W. Yasin, H. Ibrahim, N. A. W. A. Hamid and N. I. Udzir, The affects of caching in browser stage on the performance of web items delivery, *Proc. of the 2nd International Conference on Digital Enterprise and Information Systems*, Kuala Lumpur, pp.212-219, 2013.

[4] N. Markatchev and C. Williamson, WebTraff: A GUI for web proxy cache workload modeling and analysis, *Proc. of the 10th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, Washington, pp.356-373, 2002.

[5] H. ElAarag, A quantitative study of web cache replacement strategies using simulation, web proxy cache replacement strategies, *Springer Briefs in Computer Science*, London, pp.17-60, 2013.

[6] W. Yasin, H. Ibrahim, N. A. W. A. Hamid and N. I. Udzir, An overview of media streams caching in peer-to-peer systems, *The Computer Journal*, vol.56, no.7, pp.1-11, 2013.

[7] L. Boszormenyi and S. Podlipnig, A survey of web cache replacement strategies, *ACM Computing Surveys*, vol.35, no.4, pp.374-398, 2003.

[8] W. Ali, S. M. Shamsuddin and A. S. Ismail, Intelligent Naive Bayes-based approaches for web proxy caching, *Knowledge-Based Systems*, vol.31, pp.162-175, 2012.

[9] P. Cao and S. Irani, Cost-aware WWW proxy caching algorithms, *Proc. of the USENIX Symposium on Internet Technologies and Systems*, CA, USA, pp.193-206, 1997.

[10] L. G. Cardenas, J. A. Gil, J. Domenech, J. Sahuquillo and A. Pont, Performance comparison of a web cache simulation framework, *Proc. of the 19th International Conference on Advanced Information Networking and Applications*, Taipei, pp.281-284, 2005.

[11] M. Arlitt, R. Friedrich and T. Jin, Performance evaluation of web proxy cache replacement policies, *Performance Evaluation*, vol.39, no.1, pp.149-164, 2000.

[12] J. Marquez, J. Domenech, J. A. Gil and A. Pont, A web caching and prefetching simulator, *Proc. of the 16th International Conference on Software, Telecommunications and Computer Networks*, Split, pp.346-350, 2008.

[13] G. Houtzager and C. Williamson, A packet-level simulation study of optimal web proxy cache placement, *Proc. of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems*, FL, USA, pp.324-333, 2003.

[14] W. Yasin, H. Ibrahim, N. A. W. A. Hamid and N. I. Udzir, Windows web proxy caching simulation: A tool for simulating web proxy caching under windows operating systems, *Journal of Computer Science*, vol.10, no.8, pp.1380-1388, 2014.

[15] W. Yasin, H. Ibrahim, N. A. W. A. Hamid and N. I. Udzir, A systematic review of file sharing in mobile devices using peer-to-peer systems, *Computer and Information Science*, vol.4, no.1, pp.28-41, 2011.

[16] T. Hara, K. Maeda, Y. Ishi, W. Uchida and S. Nishio, Cooperative caching by clients constructing a peer-to-peer network for push-based broadcast, *Data and Knowledge Engineering*, vol.69, no.2, pp.229-247, 2010.

[17] W. Kellerer, G. Kunzmann, R. Schollmeier and S. Zöls, Structured peer-to-peer systems for telecommunications and mobile environments, *AEU – International Journal of Electronics and Communications*, vol.60, no.1, pp.25-29, 2006.