LRU-TB: Implementation of Time Based Web Caching Algorithm for E-Commerce

Mr. Navnit Singh[1],Mr. Manish Dubey[2], Mr. S.S Sekhawat[3]
[1]Department of Computer Science & engineering, AIET, Jaipur (Raj.) &
[2]Ass. Professor Department of Computer Science & engineering,AIET, Jaipur (Raj.)&
[3]Associate Professor Department of Computer Science & engineering,AIET, Jaipur (Raj.)&

ABSTRACT: This paper produced LRU-TB Web caching algorithm. Web Caching is a mechanism used to manage the bottleneck network performance by reducing network traffic, load on the Web Server and delay in retrieving the desired Web page. This is achieved by least recently used Web pages on proxy cache placed within the network. Caching can be achieved by either at client side or in the proxy server. Web proxy cache can potentially improve network performance by reducing the number of requests that reaches the server, the volume of data transferred through the network and the delay in accessing Web page. When a requested page is not available in the cache and cache is bottleneck then exchange of one or more cached documents take pace. The performance of proxy depends on page replacement algorithm. The decision by which document is evicted from the cache is depends on different kinds of replacement policies used. A lot of work is going on Web caching replacement algorithms. Different types of tools and approaches have been used by the researchers for caching. This paper illustrates design and implementation of new time based web caching algorithm for e-commerce application.It also Presents the performance analysis of the proposed system through the hit ratio and miss ratio. In future algorithm can be extended to do caching on commercial web sphere application server using http configuration

Keywords: **Web caching, frequently accessed pages, page replacement algorithm, Hit ratio, Miss ratio.**

I.INTRODUCTION

Web Caching is the temporary storage of Web objects (such as HTML documents) for later retrieval. Web Caching is a mechanism used to improve network performance by reducing network traffic, load on the Web Server and delay in accessing the Web page. This is achieved by storing frequently accessed Web pages on proxy cache placed within the network. Web proxy cache can potentially improve network performance by reducing the number of requests that reaches the server, the volume of data transferred through the network and the delay in accessing Web page. When a requested page is not present in the cache and cache is full then removal of one or more cached documents take pace. The performance of proxy depends on page replacement algorithm. The decision by which document is evicted from the cache is depends on different kinds of replacement policies used. Caching can also be utilized in the middle, between the client and the server as part of a proxy. Proxy caches are often located near network gateways to reduce the bandwidth required over expensive dedicated Internet connections. These systems serve many users (clients) with cached objects from many servers. In fact, much of the usefulness (reportedly up to 80% for some installations) is in caching objects requested by one client for later retrieval by another client. For even greater performance, many proxy caches are part of cache hierarchies, in which a cache can inquire of neighboring caches for a requested document to reduce the need to fetch the object directly. [21]

World Wide Web (WWW) is an evolving system of interlinked files like containing audio, images, videos, and other multimedia. Here Web caching is found as an important technology. Now a day's World Wide Web is widely used and this has led to a substantial increase in the amount of traffic over the internet. As a result, the Web has now become one of the primary hold up to network performance. Transferring of objects over the networks is leading to increase in the level of traffic in the network. In order to reduce such access latencies, it is desirable to replicate copies of frequently.

A. Web Caching Basic Architectures / Deployment Schemes

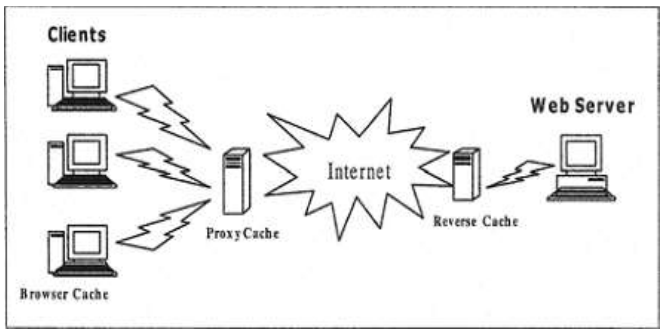

Figure 1 Web Caching on the client side, at the proxy server

Features of these three kinds of web cache can be generalized as follows:
In the Client Side: Web caches can be built into most Web browsers. A first level cache for web users is generally implemented within a browser. Because this cache only uses some of the main memory, or a small disk space for storage, the size of a browser's cache is small.
Between the Client and Server: A second level cache is usually provided within proxy cache servers using a local hard disk on the gateway server for storage. Proxy caches are often located near network gateways to reduce the bandwidth required over expensive dedicated Internet connections.[19]
In the Server Side: A reverse cache or inverse cache is placed directly in front of a particular web server. In contrast to a general proxy cache, the reverse cache only handles Web documents from one Web server. This is an attractive solution to reduce the workload of a busy web server's by caching its static documents so that the original server can be dedicated to providing service through generating dynamic pages. [22]

The paper is organized as follows: Section 2 briefly re-views the related work. In Section 3, we describe a couple of typical LRU extensions as the basis of our new replacement algorithm. Section 4 presents the LRU-TB replacement algorithms, containing accumulative cost-to-size model, implementation scheme; In Section 5, we experimentally evaluate LRU-TB with its predecessors; Section 6 describes some directions of future work and concludes this Paper.

II.RELATED WORK

A. Different Cache Replacement Algorithms: The efficiency of proxy caches is influenced to a significant extent by document placement/replacement algorithms. Cache placement algorithms try to place documents within a cache whereas cache replacement algorithms replace documents from a cache. Such algorithms are beneficial only if they can improve the hit ratio. Cache replacement algorithms often aim to minimize various parameters such as the hit ratio; the byte hit ratio, the cost of access and the latency. There are several cache replacement algorithms like **LRU (Least Recently Used)[12], LFU (Least Frequently Used)[21], SIZE[20], GDS (Greedy Dual Based)[13], LRV (Least Relative Value)[16], Smart Web Caching[22].**
The common findings after the review of research papers are listed as:

- Client-side caching refers to caches that are built into most web browsers, which caches Internet objects for a single user, but from a variety of servers.
- K-mean algorithm is applied for user data personalization. K-Means algorithm could measure dissimilarity between two objects with the distance. When K was given, they were randomly assigned all the objects to the non-empty K clusters, and then calculated the average of each cluster.
- The Size-Adjusted LRU sorts all objects in cache in terms of the cost-to-size ratio and Segmented LRU is a frequency-based extension to basic LRU especially designed for disk caching where all pages are identical in size.
- Multi-level cache generally operated by checking the smallest level (L1) cache first. If miss occur in smaller cache than next larger level cache (L2) is checked.
- Lowest Relative Value (LRV) Algorithm is based on maximizing an objective function for the whole cache. The objective function used a cost/benefit model to calculate the relative value of each document in the cache.
- Least Recent Used (LRU) cache replacement algorithm used to replace data object that have not been accessed longest time.
- In trace-driven simulation, a trace of the memory references executed is created, usually either by simulation or by instrumented execution. The trace includes what instructions were executed (given by the instruction address).
- Cache Listener is mainly used to response the requests from the client, which provides two RESTful (Representational State Transfer) Services – the POST and GET. When receiving a GET/POST request, it formats the received data for the LRU Cache Model and sends the data stream to the client after the LRU Cache computes the data.
- Event Driven Simulator allows the exact modeling of the interaction between the two, which is critical. In this simulator, when the program is executed performance study is performed at the same time.
- LFU algorithm is based on object size. It's replacing data objects that have been accessed least frequently. LFU evicts the least frequently accessed document first, on the basis that a popular document tends to have a long-term popularity profile.

B. Problem Definition And Objectives

The main aim of the paper is to generate domain-specific answers of the user entered query. To achieve the goal of the "Implementation of Web Caching Replacement Algorithm for E-Commerce" following objectives are framed:

- To study and analyze existing LFU algorithm
- To design and implement the modified LFU algorithm.
- To validate the proposed algorithm through development of GUI based tool.

Caching Web objects at locations close to the user has been accepted as one of the solutions to Web server bottlenecks which reduce traffic over the internet and improve the scalability of the system. Caches act as intermediate systems that interrupt the user requests before they arrive at the remote server. A Web cache checks if the requested object is available in its local cache, if it's available then cached page is sent back to the user; otherwise the cache redirect the request to the origin server. When the cache receives requested object it store it in local cache and forwards back the results to the user. The copies kept in the cache are used for subsequent users' requests. Web proxy cache can potentially improve network performance by reducing The number of requests that reaches the server, The volume of data transferred through the network and The delay that an end-user incurred while accessing Web page.

III.LRU EXTENSION WORK

A.Architecture of Web Cache Replacement algorithm and Its Assumption

This chapter discusses the system process flow of the work, architecture of the proposed Cache Replacement Algorithm, methodology and Simulation Tools used.

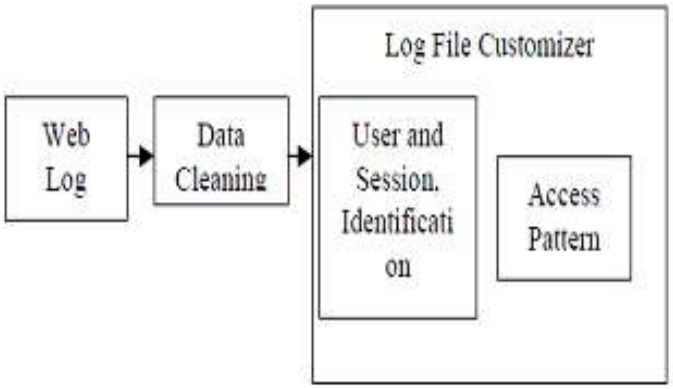

Figure 2.Existing Work

The processes involved in generating Web access pattern from Web log are:

- Data Cleaning
- Identifying frequent user and pages
- Access pattern generation

Data Cleaning: The proxy server log file processor processes the raw proxy server log file to obtain a processed log file. All the proxy server logs requests that refer to the image, CSS, and java script files are removed from the Web log file. Those entries like IP address, Web pages and its size are requested by the end users are retained in the processed log file.

Log File Customizer: Log File Customizer uses the data obtained from the cleaned proxy log file for sampling. A sub module of the log file customizer includes user identification and session identification. In user identification unique users are identified based on IP address and stored in one dimensional array Frequent User. In session identification for each user it identifies the set of URL requested by particular user during the predefined time period. Once users and pages is identified then frequent user and frequent page is identified by counting total number of visits. Here threshold value of is set dynamically used. It includes the following sub modules:

- Frequent user identification
- Frequent page identification
- Access Pattern generation

B.Frequent User Identification Algorithm: This algorithm used for identify the frequent user for the web application based on customer id and page id. In this algorithm, Page IDs, Page Names and User Ids taken as input. It retrieves the Page IDs and if the selected page ID is valid then retrieves the page count of the page for particular user. If the page count is greater than threshold then it is the frequent user for the given page id.

Algorithm: Frequent User Identification

Input- User IDs and Page Names

Output – Frequent User

Step 1: Page Count and user id is store in log file.

Step 2: For every login page count is increment

Step 3: Set Threshold value.

Step 4: if (Page Count >= threshold value) then

Step 5: Frequent user identified and the page count incremented by 1 and the store the database.

Step 6: Otherwise increment the page count incremented by 1 and the store the database.

Step 7: Stop

***C.Frequent Page Identification Algorithm*:** This algorithm used for identify the frequent page for the web application based on his customer id. In this algorithm, Page IDs, Page Names and User Ids taken as input. It retrieves the List of Pages, if the Page count is greater than Threshold for the user. Then the page becomes the frequent page.

Algorithm: Frequent Page Identification
Input- Page IDs and Page Names, Customer Id
Output – Frequent Page
Step 1: Page Count and Page id is store in log file.
Step 2: For every login page count is increment
Step 3: Set Threshold value.
Step 4: if (Page Count >= threshold value) then
Step 5: Frequent page identified and the page count incremented by 1 and store.
Step 6: otherwise increment the page count incremented by 1 and the store the database.
Step 7: Stop

***D.Access Pattern Generation*:** This algorithm used for identify the number of times a page for the web application is visited by a specific customer id. In this algorithm, Page IDs, Page Names and User Ids taken as input.

Algorithm: Access Pattern Generation
Input- Page id, User Id
Output – Access Pattern on Particular page for different user
Step 1: User Id and Page id is store in log file.
Step 2: Calculate page count for every login
Step 3: For every login page count is increment and
Goto step 5
Step 4: Access Pattern Generate and store in Database
Step 5: Stop

E.Least Frequently Used (LFU) Algorithm: This cache system is used to find the priority of a page and render the page from the cached system if it exceeds threshold. Each time user does some action or navigates to the page then the count is incremented against that page. This Cache system is the settings which has 3 items CACHEON indicates whether the LFU is ON or OFF. Threshold indicates the frequent page threshold for a specific user. The count of the page is determined for the user i.e. it exceeds the threshold then it gets the redirected to Help URL.

Algorithm: Least Frequently Used
Input- User IDs, Page Id and Threshold Value
Output – Cache Replacement for frequent user
Step 1: Page Count and user id is store in log file.
Step 2: Page requested by customer id
2.1. If Page ID found into database for this user then Increment page count by 1 and store into database
Goto Step 3
2.2. Else Initialize Page Count=1 and store in database
Goto Step 3
Step 3: Page Count Information for user is retrieved from database
Step 4: Set Threshold value
4.1. If (Page count>= threshold) then Frequent user identified based on page count and **Goto** Step 5
4.2. Else **Goto** Step 6
Step 5: Connect the user on application server 2.
Step 6: Connect the user on application server 1.
Step 7: stop

IV.IMPLEMENTATION

A. Architecture of Proposed work

The Architecture the proposed system is as shown in the figure 2. The key components in the scheme are the proxy server log file pre-processor, customized log file processor, and access pattern generation. In the log file we have recorded the time stay in particular page. Based on log file customizer, frequent user transferred on second server in proposed scheme.
In this proposed work admin set the threshold. Based on that threshold compare the stored time in database and user redirect to second server.

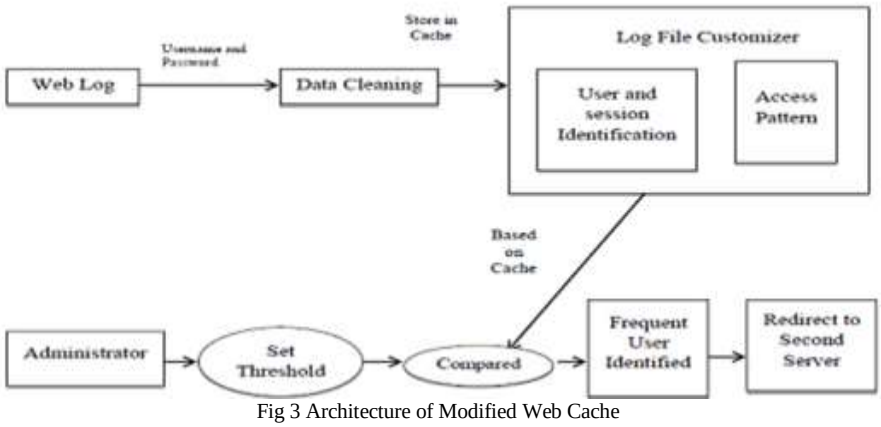

Fig 3 Architecture of Modified Web Cache

B. Algorithm for the Proposed Web Caching System based on Page Visit Time:

This cache system is used to find the priority of a page and render the page from the cached system if it exceeds threshold based on the timing information and frequent page threshold.

Algorithm: **Modified Page Replacement Algorithm**

Input- User IDs and Page Id, Threshold Value, page visit time

Output – Frequent User with alternate server based on time

Step 1: Page Count and user id is store in log file.

Step 2: Page requested by customer id

2.1. If Page ID found into database for this user then Increment page count by 1 and store into database Goto Step 3
Else Initialize Page Count=1 and store in database Goto Step 3

Step 3: Page Count Information for user is retrieved from database

Step 4: Compute page visit time for particular page and store in database

Step 5: Set Threshold value

5.1. If (Page count>= threshold) then Frequent user identified based on page count and Goto Step 6
Else Goto Step 7

5.2. If (page visit time>= Threshold) then Frequent user identified based on time stay and Goto Step 6
Else Goto Step 7

Step 6: Connect the user on application server 2

Step 7: Connect the user on application server 1.

Step 8: stop

C. Parameters for Measuring Web Caching Performance:

To find the efficiency of targeted work, its performance is checked by measuring the Hit Ratio and Miss Ratio of cache. Generally, various number of performance parameters are available, but in the proposed work Hit Ratio and Miss Ratio of cache is used to measure the performance of the system.

Cache Hit Ratio: The hit rate is generally a percentage ratio of documents obtained through using the caching mechanism versus the total documents requested. In others words, the ratio of the number of requests met in the cache to the total number of requests is called hit ratio.

$$Hit\ Ratio = Total\ no\ of\ request\ met\ in\ cache/Total\ no\ request$$

Cache Miss Ratio: The miss ratio is the fraction of accesses which are a miss. It holds that

Miss Ratio = 1-Hit Ratio

V. RESULT AND ANALYSIS

This chapter discusses about experimental results and analysis. In the experiment, First we have created web based application which contains two panels: first is admin panel and second is user panel. In user panel manually created four pages for caching. And the admin panel considered Frequency algorithms, setting of LFU and proposed Modified LFU, performance measurement etc.

First phase is registration phase in which user is added in database. After successful completion success message appear.

Second phase is login phase in which user login to E-commerce website and perform task like searching, buying etc.

Figure 4 E-commerce page

A.Frequent User Identification

In the frequent user identification algorithm, initially admin set the threshold value for frequent user. When user request the pages it count the page and increment page count by 1 for each request for the specific user. If page count is greater than threshold for particular user it retrieves as the frequent user as is shown in table 1.

Table 1.Threshold Setting

INPUT	OUTPUT(Frequent User)	
Threshold Set by ADMIN(for ex 5 set by admin)	No of times customer Visited	Customer ID
	7	aaaaa
	4	Ab
	9	Navnit
	2	acad

In the upper Table admin set the threshold value 5 for identify the frequent user. In this table admin take threshold input 5. The entire user which visits 5 times on the pages of website is shown in above table.

In the frequent page identification algorithm, firstly admin set the threshold value. In the frequent page identification identify that which page visited more time as a frequent page.

B.Access Pattern Generation

Access pattern show all the information of user and page, like as the number of times a page for the web application is visited by a specific customer id. This is shown in table 2.

Table 2. Time Informtion

ID	No of Time Page Visited	Page Name	Customer Id
311	1	SHOP1	nav
312	2	LATEST BUYED	ram
313	1	SHOP1	sk
314	2	LATEST BUYED	navnit
315	2	LATEST BUYED	Vk
316	1	SHOP1	Vk
317	2	SHOP1	Ab

B.Experimental Setup

Cache Setting:

In the cache setting two methods LFU setting and MLFU setting are implemented. These setting used for finding the frequent user based on page count and time stay of a page respectively. And according to frequent user identification user are connected to the second server.

LFU Cache Setting: The LFU cache only considered user as frequent user, whose page count >= Threshold. In the LFU cache setting, has 3 items CACHEON column indicates whether the LFU is ON or OFF. Threshold indicates the frequent page threshold for a specific user. Since the Threshold is 10.The count of the page is determined for the user; if it exceeds the threshold then it gets the redirected to Help URL as shown in Table 3.

Table 3.LFU Cache setting

CACHEON	THRESHOLD	CACHE ID
1	5	1

LRU-TB Cache System

The MLFU cache considered user as a frequent user where page count >= Threshold OR page_visit_time>= Threshold. In the MLFU cache setting, has 3 items CACHEON=1 or 0, THRESHOLD=100 and CACHEID=1. If CACHEON=1 that means that MLFU cache is ON. The Threshold is 100 if the page exceeds the threshold 100 then page will get accessed through the second application server as shown in Table 4.

Table 4.LFU-TB Cache Setting

CACHE ON	THRESHOLD	CACHE ID
1	100	1

For experiment purpose 10 users visited on different pages are created, as shown in table 4.Those ID's time of stay is recorded in the database table. For evaluation of the experimental results, Hit Ratio and Miss Ratio are used. The values of the above performance measures are calculated, for LFU and MLFU in proposed systems and compared.

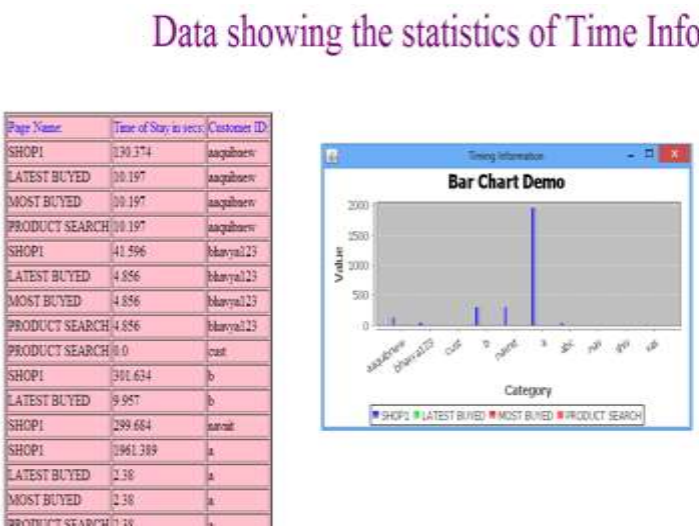

Figure 5.Time Information in GUI

Based on the time information and HIT,MISS Ratio frequent user is identified. If user is frequent then it is redirect to second server.

Figure 6.Redirection to second server

VI. Conclusion And Future Scope

The review of 30 research papers has been carried out in the area of Web caching and replacement policies and found current challenges and scope of the work in the area. After the review, an issue was found in the literature which was based on experimental approach. Study of literature was carried out in depth of common finding of research works, strength and weaknesses and gaps to build problem statement and objective.

- To implement the log file customization for different user.
- To design the scenarios of existing LFU algorithm
- To design and implement the modified LFU algorithm to find the frequent user.
- To design and implement the 2nd server module for frequent user.
- The algorithms can be extended to do caching on commercial web sphere application server using HTTP conf file.

In the proposed work we have used Page Count and Timing information for finding the frequent user. There are some problems also found out, this problem are discussed below. The algorithms can be extended to do caching on commercial web sphere application server using HTTP conf file. Web caching is used to reduce server workload by storing data in proxy caches that is placed over the network.

REFERENCES

[1]. A. Bhattacharjee and B. K. Debnath, "**A new Web cache replacement algorithm,"** *PACRIM. 2005 IEEE Pacific Rim Conference on Communications, Computers and signal Processing, 2005*, pp. 420-423.

[2]. A.Gao, D. Mu, H. Su and W. Pan, "**Proportional Hit Rate in Caching Service: A Feedback Control Approach**," *International Symposium on Computer Network and Multimedia Technology*, Wuhan, 2009, pp. 1-4.

[3]. A. Horiuchi and K. Saisho, "**Prototyping and Evaluation of Virtual Cache Server Management Function for Distributed Web System**," *International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, 2015, pp. 324-329.

[4]. A. Sarhan, A. M. Elmogy and S. M. Ali, "**New Web cache replacement approaches based on internal requests factor**," *9th International Conference on Computer Engineering & Systems (ICCES)*, Cairo, 2014, pp. 383-389.

5. A. Singh and A. K. Singh, "**Web Pre-fetching at Proxy Server Using Sequential Data Mining**," Third International Conference on Computer and Communication Technology, Allahabad, 2012, pp. 20-25.
6. Dr.K.Ramu1 and Dr.R.Sugumar, "**Design and Implementation of Server Side Web Proxy Cache Algorithm**," international journal of advanced Research in computer and communication engineering *VOL. 1, ISSUE 1, MARCH 2012.*

7. H. K. Yogish and G. T. Raju, "**A Novel ART1NN Clustering and Pre-fetching Technique for Reducing Web Latency**," *5th International Conference and Computational Intelligence and Communication Networks*, Mathura, 2013, pp. 20-25.

8. Hwangcheng Wang, JunzhiPeng, Yuting Wu and HsiantaiFeng, "**SzLFU(k) Web cache replacement algorithm**," *TENCON '02. Proceedings. IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol.2, 2002, pp. 20-25.
9. Strategy of Web Cache Based on Data Mining,"10th International I. S. Sette et al., "Analysis of Prediction and Replacement Algorithms Applied to Real Workload for Storage Devices," IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Washington, DC, 2012, pp. 507-509.

10 .J. Zhang, "Replacement Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Krakow, 2015, pp. 821-823.

11.K. Geetha, N. A. Gounden and S. Monikandan, "SEMALRU: An Implementation of modified web cache replacement algorithm," World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, 2009, pp. 1406-1410.

12.Kai Cheng and Y. Kambayashi, "LRU-SP: a size-adjusted and popularity-aware LRU replacement algorithm for web caching," Proceedings 24th Annual International Computer Software and Applications Conference. COMPSAC2000, Taipei, 2000, pp. 48-53.

13.M. Busari and C. Williamson, "On the sensitivity of Web proxy cache performance to workload characteristics," Proceedings IEEE INFOCOM. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213), Anchorage, AK: 10.1109/INFCOM.2001.916617, 2001, pp. 1225-1234 vol.3..

14.M. S. A. Khaleel, S. E. F. Osman and H. A. N. Sirour, "Proposed ALFUR using intelegent agent comparing with LFU, LRU, SIZE and PCCIA cache replacement techniques," International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE), Khartoum, 2017, pp. 1-6.

15.M. Song, H. Li and H. Wu, "A Decentralized Load Balancing Architecture for Cache System," International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Xi'an, 2015, pp. 114-119.

16.N. Bian and H. Chen, "A Least Grade Page Replacement Algorithm for Web Cache Optimization," First International Workshop on Knowledge Discovery and Data Mining (WKDD 2008), Adelaide, SA, 2008, pp. 469-472.

17.P. Bangar and K. N. Singh, "Investigation and performance improvement of web cache recommender system," International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), Noida, 2015, pp. 585-589.

18.Q. Li, X. Liao, H. Jin, L. Lin, X. Xie and Q. Yao, "Cost-Effective Hybrid Replacement Strategy for SSD in Web Cache," IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Liverpool, 2015, pp. 1286-1294.

19.S. M. Abid and H. Youssef, "Impact of One-Timer/N-Timer Object Classification on the Performance of Web Cache Replacement Algorithms,"

20.IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Toronto, ON, 2010, pp. 208-211.

21.Seung-Won Shin, Ki-Young Kim and Jong-Su Jang, "LRU based small latency first replacement (SLFR) algorithm for the proxy cache," Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003, 2003), pp. 499-502.

22.V. Sathiyamoorthi and V. MuraliBhaskaran, "Web caching through modified cache replacement algorithm," International Conference on Recent Trends in Information Technology, Chennai, Tamil Nadu, 2012, pp. 483-487.
23.W. Hui-chang, R. Shu-hua and T. Qi-jie, "The Implementation of a Web Crawler URL Filter Algorithm Based on Caching," Second International Workshop on Computer Science and Engineering, Qingdao, 2009, pp. 453-456.
24.WengMeizhen, Shang Yanlei and TianYue, "The design and implementation of LRU-based web cache," 8th International Conference on Communications and Networking in China (CHINACOM), Guilin, 2013, pp. 400-404.

25.X. Meng, C. Si, X. Han, J. Zhang and L. Xu, "A Replacement Algorithm Designed for the Web Search Engine and Its Application in Storage Cache," IEEE International Symposium on Parallel and Distributed Processing with Applications, Chengdu, 2009, pp. 53-59.

26.X. Zou and C. Chen, "HQ: An Architecture for Web Cache Replacement Algorithms in Distributed Systems," International Conference on Computer and Communication Engineering (ICCCE), Kuala Lumpur, 2016, pp. 78-83.

27.Y. Niranjan, S. Tiwari and R. Gupta, "Average memory access time reduction in multilevel cache of proxy server," 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad, 2013, pp. 44-47.

28.Y. Zhang, D. Li and Z. Zhu, "A Server Side Caching System for Efficient Web Map Services," International Conference on Embedded Software and Systems Symposia, Sichuan, 2008, pp. 32-37.

29.Z.Xiong, W. He and N. Liang, "A PnP Scheduling Algorithm and Cache Replacement Algorithm Simulation Platform for Web Server Cluster,"Fourth International Conference on Computational and Information Sciences, Chongqing, 2012, pp. 888-891.

30. Z. Zeng and B. Veeravalli, "Hk/T: A Novel Server-Side Web Caching Strategy for Multimedia Applications," IEEE International Conference on Communications,Beijing,2008,pp.1782-1786.