VOL. 12, NO. 3, FEBRUARY 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

COMBINATION OF FIFO-LRU CACHE REPLACEMENT ALGORITHMS ON PROXY SERVER TO IMPROVE SPEED OF RESPONSE TO OBJECT REQUESTS FROM CLIENTS

Tanwir[1, 2], Gamantyo Hendrantoro[1] and Achmad Affandi[1]
[1]Department of Electrical Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
[2]Department of Electrical Engineering, Universitas Sains dan Teknologi Jayapura, Jayapura, Indonesia
E-Mail: tanwir@ieee.org

ABSTRACT

In this paper, cache is the repository of browsing results located in cache disk. The size of cache repository and the choice of cache replacement algorithm affect the speed of a system. Improper deletion of an object during cache replacement may erase the most frequently used objects and cause misses during request. In this study, we propose a method of throughput improvement by combining FIFO (First in First Out) and LRU (Least Recently Used) cache replacement algorithms. The analysis was conducted to identify the effect of cache size on hit rate percentage, response time, delay time, and throughput when the combined FIFO-LRU algorithm is applied. The finding indicates bandwidth efficiency improvement compared to single algorithms, as showed by 73% throughput improvement on 200 MB cache. The application of the combined algorithm also reduces bandwidth usage and delay time while minimizing miss rate and increasing hit rate.

Keywords: cache size, throughput, FIFO, LRU, FIFO-LRU, miss, hit rate.

1. INTRODUCTION

In computer network, throughput is the measure of data flow speed indicating the amount of actual information flowing from one point to another within certain period, expressed in bits per second (bps) [1], [2]. In the context discussed in this study, the data bits are stored inside cache on proxy server [3]. Inside a terminal accessing the internet, browsers such as Internet Explorer, Firefox, and Opera store web documents in forms of HTML files, images, and video [4] and audio streaming of the visited webpages using caching technology. The application of caching technology aims to minimize bandwidth usage [5] [6] and proxy server load so that the browsers can download the object faster when revisiting the webpages in the future. Caching also helps download multiple objects by directly copying the objects from directories where the cached files are being stored [7] [8]. To clarify, we need to differentiate disk cache from memory cache. Disk cache is the repository of caches located in proxy server used to store the requested objects while memory cache is located between registers and main memory inside the CPU. This study will focus on disk cache.

The function of cache replacement algorithm is to delete the stored objects in the cache based on specific criteria. Cache replacement algorithm does affect the speed of a system because it may replace/delete the objects that are commonly used that result in "miss"during request. There are several cache replacement algorithms with their own advantages and disadvantages [9]. Least Recently Used (LRU) algorithm replaces the least accessed object without any references, while First in First out (FIFO) algorithm replaces the earliest stored object earliest. The advantage of LRU lies in its ability to replace the least accessed objects without considering when the object is stored in the disk cache for the first time. Meanwhile, the advantage of FIFO algorithm is its ability to replace the oldest stored objects with the new ones without considering when the last time the objects being accessed.

"Hit" refers to a condition when a system finds and displays the results of requested objects in a cache [10] [11],while a condition when the system fails to find the requested objects is called "miss" [12] [13] [14]. The probability (commonly expressed in percentage) of finding the requested object by a system is called hit rate, while the percentage of system failure in finding the requested object is called miss rate [15] [16]. The configuration of objects on cache replacement depends on the management of the applied algorithm. In request using a single LRU or FIFO algorithm, an object (or several objects) may be deleted during cache replacement. For example, in FIFO algorithm, the oldest saved object will be deleted earliest and the objects that may be still used are deleted, causing miss and delay. In this condition, LRU algorithm seems to be able to maximize hit rate. However, the advantage of FIFO algorithm lies on its cache replacement efficiency because of its queue system according to the sequence of stored object. Unlike in LRU algorithm [17], stack rarely occurs with FIFO algorithm. FIFO algorithm is static and easily combined with other algorithm. Therefore, by combining FIFO algorithm and LRU algorithm, we expect the combination of their respective advantages, resulting in maximum hit rate, minimum miss rate, and efficient cache usage.

Hence, the contribution of this study is the evaluation of the combination of the two cache replacement algorithms to minimize missrate. Specifically, we suggest the use of FIFO-LRU combined algorithm to optimize the advantages of each algorithm. The result of evaluation indicates that the combined FIFO-LRU

algorithm is able to increase the throughput compared to single algorithm.

Section II below discusses types of cache replacement algorithms individually, including the single and the combined algorithms. In Section III, we explain the experiments aimed to evaluate the performance of each algorithm during request. Finally, in Section IV we present the conclusion of this study.

2. CACHE REPLACEMENT ALGORITHMS

A computer system consists of CPU, RAM, HDD, and others. Memory cache is a small-sized high-speed memory located in the CPU used to store the copies of objects and instructions accessed by the CPU. Memory cache also functions to bridge the difference between CPU speed and main memory. However, this study focuses on disk cache located in proxy server [18] (as illustrated in Figure-1 below). The main function of disk cache located in proxy server is to store the objects in the internet accessed by the users. Therefore if a user request internet service containing the objects accessed or requested previously (i.e. the objects exist in the cache) the proxy service will directly find and display the objects to the user without re-requesting the objects to the internet server.

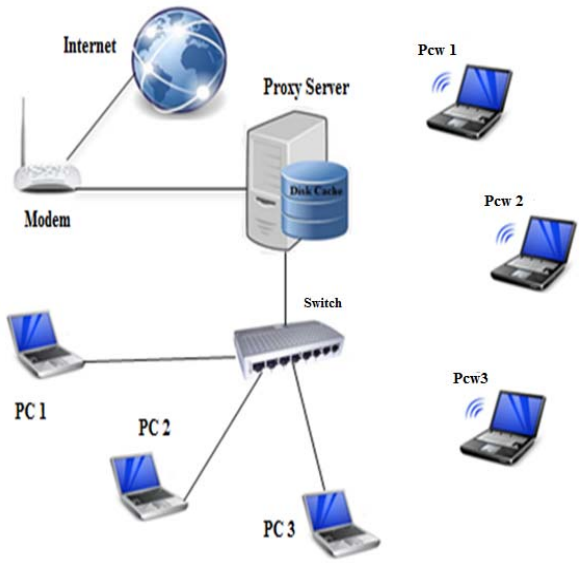

Figure-1. Illustration of a network with clients accessing the Internet using Proxy Server.

The basic working principle of a cache is determining whether a request on certain objects has been previously made. In general, this process conducted by matching a new request to the existing similar queries. If a similar query exists or has been made before, a cache hit occurs and the system finds and displays the results of searching to the user. On the other hands, if the similar query does not exist or never made, a cache miss occurs and the system requests the results to internet server. In the following sub-sections, we discuss single FIFO and LRU algorithms and the new replacement algorithm combining both FIFO and LRU as the contribution of this study. The pseudo-codes for the algoritms are given for a scenario with requests coming from three laptops as the objects of this study, defined respectively in the forms of Xn, Yn, and Zn.

a) Least Recently Used (LRU) algorithm: Least Recently Used (LRU) algorithm works by deleting the objects that are not used recently. The idea is that the request on the objects that are not recently used are least possible so that other newer objects that have higher possibilities to be requested can use the space. LRU algorithm implements counter logical clock and stack logical clock. In counter logical clock, every object has initial score 0 (zero). Every time an object is being accessed, the score increases and the system will replace the object with the lowest score. Meanwhile, in stack logical clock, the recently accessed objects are put on the top of the stack and the most bottom object will be replaced. In Table-1 presenting the pseudo-code of LRU algorithm, the implementation of counter and stack logical clocks were accomplished during the evaluation of disk cache expiry.

Table-1. LRU Algorithm Pseudo-code.

```
Read Xn; Yn ; Zn
If (disk cache Expires) then,
Print "disk cache"
If (New Object) then,
Print "disk cache"
Else
Print "disk cache LRU"
End If
```

b) First in First out (FIFO) algorithm: FIFO algorithm replaces the oldest object stored in cache. The algorithm assumes that the oldest objects will not be requested anymore so that the objects may be replaced with the new ones. The disadvantage of this algorithm is that it deletes the objects that are considered as expired although the objects might still be actively used. FIFO applies logics that the first stored objects will be deleted first, ass implemented by the pseudo-code in Table-2.

Table-2. FIFO algorithm pseudo-code.

```
Read Xn; Yn; Zn
If (disk cache First In) then,
Print "disk cache"
If (New Object) then,
Print "disk cache"
Else
Print "disk cache FIFO"
End If
```

c) FIFO-LRU combined algorithm: The work mechanism of a cache is to determine whether requested objects exist in the cache or not. When a requested object exists, a cache hit occurs. On the other hand, if the

requested object does not exist, a cache miss occurs and the system will search for the requested objects in the network. The combination of FIFO and LRU algorithms is expected to improve the efficiency of storing and requesting objects on the cache because the algorithm does not delete the expired objects that may be actively used in the future.

Table-3. Combined LRU-FIFO algorithm pseudo-code.

```
Read Xn; Yn ; Zn
If (disk cache full ≥ CacheSize then,
Print "disk cache"
If (fetch count ≥ MaxCount; expires≥ MaxTime) then
Print "disk cache"
Else
Print "disk cache FIFO-LRU"
End If
```

In FIFO-LRU pseudo-code presented in Table 3 above, CacheSize, MaxCount, and MaxTime serve as parameters of the algorithm. By combining FIFO and LRU algorithms, the often requested objects will not be deleted although in terms of cache occupation time the objects should have been expired. On the other hand, the combined algorithm is expected to be able to increase the efficiency of cache usage

Cache functions as temporary storage. When a request on certain object occurs, the system will search the object in cache. If the system cannot find the requested object in cache, the processor continues searching in RAM with slower speed. A disk cache provides the data required by the processor and the effect of slow-speed RAM performance can be minimized. Through this mechanism, bandwidth increases and the processor work more efficiently. Higher capacity disk cache may improve overall computer performance.

3. EVALUATION

The performances of the observed algorithms are measured according to several criteria, namely hit rate, response time, delay time, and throughput. Hit ratio is measured through the ratio between the number of requests on certain objects accepted on cache and the number of objects sent back by the client to the cache. Response time refers to time required by the client to request a certain object (whether the object exists in the cache or not) until the client receives responses related to the request. Delay time is defined as total time required finding an object in the disk cache, starting from requesting the object until the search finishes (the result is either a hit or a miss). Throughput is measured as the ratio of hitobjects expressed in bits to the observation time.

The evaluation on hit rate and response time as the performance parameters of FIFO algorithm, LRU algorithm, and the combined FIFO-LRU algorithm was conducted by sending various different objects to the request system. As described in Figure-2 below, several requests on certain objects were made during evaluation. Cache miss occurred when the cache could not complete the request while cache hit occurred when the cache completed the request. We also carried out cache validation, a process assuring that the system does not provide expired data to the client. Proxy server validation runs according to the mechanism that it will only send back the copy of objects that have not expired to the client.

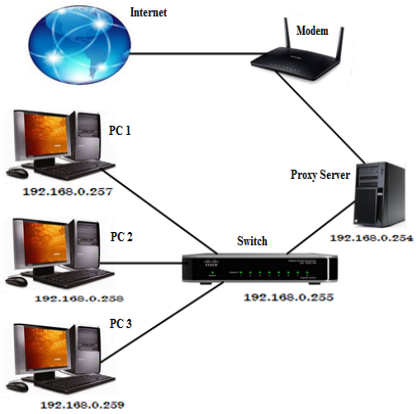

Figure-2. Configuration of network system used during evaluation.

During the experiments requests on different objects Xn, Yn, and Zn were made from three devices (laptops) accessing the internet simultaneously. The combined FIFO-LRU cache replacement algorithm indicates good and fast response that increases hit ratio percentage on cache size. The experimental results are given in the forms of graphics describing hit rate percentage on cache size (Figure-3a), response time on cache size (Figure-3b), delay time on cache size (Figure-3c), and throughput on cache size (Figure-3d).

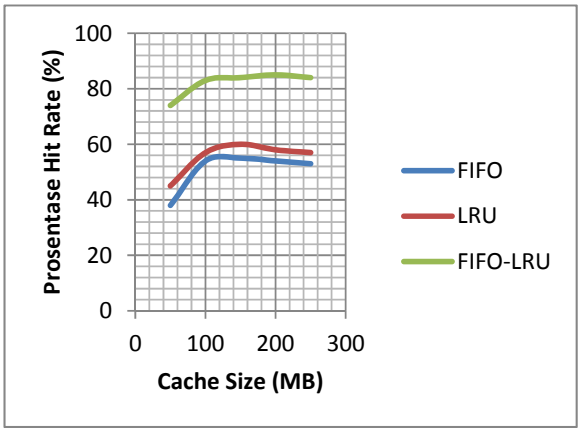

Figure-3(a). Hit rate percentage vs cache size.

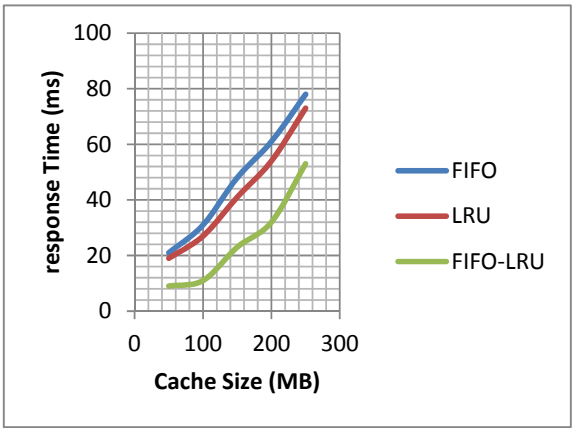

Figure-3(b). Response time vs cache size.

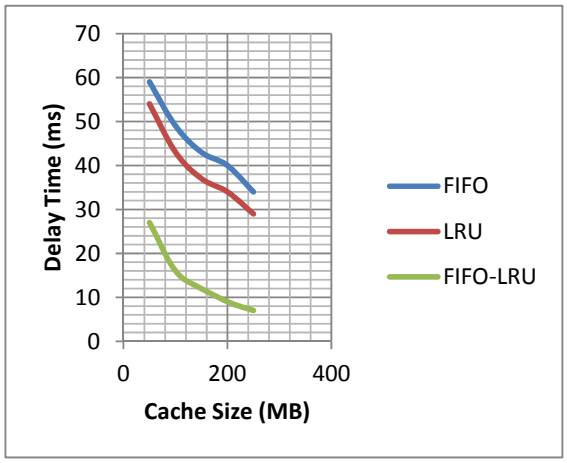

Figure-3(c). Delay time vs cache size.

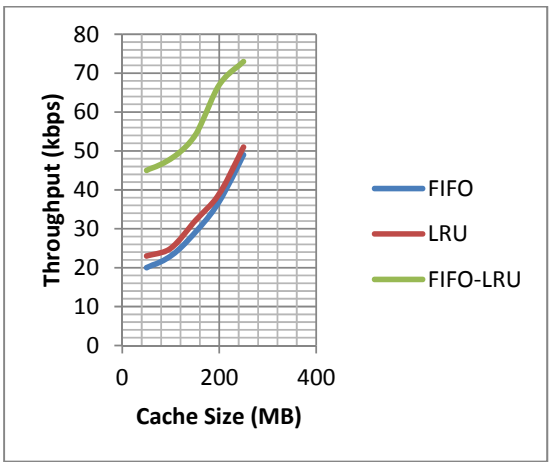

Figure-3(d). Throughput vs cache size.

Figure 3(a) above indicates that hit rate increases with the increasing cache size. Hit rate percentage increases with the increasing cache size up to 100 MB. Yet after cache size reaches 100 MB, hit rate percentage tends to be constant. The increasing number of requests executed by the cache (hit) is indicated by the increasing percentage of object taken from the cache relative to the number of request sent by the cache to the client. This condition appears because a number of objects saved in the cache applying single FIFO or LRU algorithm were deleted, whereas in combined FIFO-LRU algorithm there is no object deletion if there are many requests on them.

For example, the result of initial measurement of request on certain object with cache size 50 MB the percentage of hit rate on total request was 38% using FIFO algorithm and 45% on LRU algorithm. Meanwhile, implementation of FIFO-LRU algorithm provides the highest hit rate percentage as much as 73% which increases up to 85%.

Figure-3(b) shows that the response time increases with the increasing cache size. The result of initial analysis towards response time of the evaluated algorithms on 50 MB cache size indicates that the response times for FIFO, LRU, and combined FIFO-LRU algorithms are 21, 19 and 10 milliseconds, respectively. The response time of the algorithms also increases as the cache size increases to 100 MB, in which the response times of FIFO, LRU and FIFO-LRU algorithm are 32, 27, and 11 milliseconds, respectively. Similarly for 250 MB cache size, the response times are 78, 73, and 52 milliseconds. The explanation of this condition is that the elapsed time for data query will be longer on larger cache size.

Figure-3(c) depicts the delay time of each algorithm in different cache sizes. In 50 MB cache size, the delay times of FIFO, LRU and FIFO-LRU are 59; 53; and 26 milliseconds, respectively. Also similarly for 150 MB cache size, the delay times are 42; 37; and 12 milliseconds, respectively. From these findings, there is a phenomenon in which the increasing cache size reduces miss rate, causing a decrease in delay time, which in turn results in faster object finding.

The implementation of combined FIFO-LRU algorithm significantly reduces delay time (as indicated on Figure 3c above). As the cache size increases, it takes longer to perform a request. After the request becomes cache hit, the object will be stored in the disk cache reducing the delay time of future requests on the same object. When the disk cache repository is full, the FIFO-LRU algorithm shall delete the objects according to counter logical clock and stack logical clock on the queued objects. The deletion significantly reduce delay time with cache size.

Figure-3(d) shows that the throughput reached on 50 MB cache size is 20 Kbps by FIFO algorithm; 22 Kbps by LRU; and 45 Kbps by FIFO-LRU. As the cache size increases to 150 MB, the throughput of the algorithm also improves. FIFO algorithm improves to 28 Kbps; LRU improves to 32 Kbps; and the combined FIFO-LRU improves to 54 Kbps. Throughput improvement increases the possibility of finding the requested objects within the cache and reduces access time to the server.

The comparison of the hit rate percentage, response time, delay time, and throughput among the three algorithms indicated that LRU algorithm always has a slightly better performance than FIFO. This phenomenon

is consistent to the concept that LRU keeps the most accessed objects within the cache, one condition that cannot be done by FIFO algorithm on the expired objects. However, the combination of FIFO and LRU algorithms has significantly better performance on almost all criteria compared to the single algorithms. For example for cache size 100 MB, the response time of LRU is 4 milliseconds faster than FIFO and 6 milliseconds faster for 200 MB cache size. Meanwhile, the combined FIFO-LRU algorithm response time is much faster than the single algorithms. The combined FIFO-LRU algorithm is faster than LRU algorithm by 15 milliseconds on 100 MB cache size and by 20 milliseconds on 200 MB cache size.

The application of LRU algorithm, which is on stack and counter logics, may cause skips when the objects are replaced during cache replacement process that increases delay time. On the other hand, the basic principle of FIFO is the oldest input will be replaced earlier without considering whether the objects are actively used or not on request to the client server. If we combine LRU algorithm with FIFO algorithm implementing queuing principle, the objects that are originally skipped in the single algorithms will follow queuing principle. By applying combined FIFO-LRU algorithm, a number of hit requests for objects readily saved in cache do not require further queries, thereby reducing searching time and improving response time.

The application of combined FIFO-LRU relatively improves the throughput relative to LRU algorithm by 85% on 100 MB cache size and by 73% on 200 MB cache size. Throughput improvement affects bandwidth usage efficiency and reduces delay time to 27% with respect to the LRU algorithm.

4. CONCLUSIONS

This paper has proposed the application of a cache replacement algorithm that combines LRU and FIFO single algorithms. The combination of FIFO and LRU algorithms aims to combine the strengths of both single algorithms so that the combination can improve object access performance based on several criteria, namely hit rate, response time, delay time, and throughput.

General results of experimental evaluation indicate that the application of combined FIFO-LRU algorithm brings relatively great improvement performance, compared to single LRU algorithm. The hit rate percentage of FIFO-LRU algorithm increases to 73% for 200 MB cache size and increases to 85% on 100 MB cache size. As the hit rate percentage improved, the request speed increases and response time decreases. As a result, the delay time also decreases, resulting in shorter response time. As the hit rate increases, time required to find an object during future requests will be shorter, thereby causing the throughput to increase. Based on these findings, the application of combined FIFO-LRU is strongly recommended.

REFERENCES

[1] M. Ji, G. Caire and A. F. Molisch. 2015. The Throughput-Outage Tradeoff of Wireless One-Hop Caching Networks. IEEE Trans. Inf. Theory. 61(12): 6833-6859.

[2] H. Ajorloo and M. T. Manzuri-Shalmani. 2016. Throughput Modeling of Distributed Reservation Protocol. IEEE Trans. Mob. Comput. 15(2): 503-515.

[3] W. Ma and D. H. C. Du. 2004. Design a progressive video caching policy for video proxy servers. IEEE Trans. Multimed. 6(4): 599-610.

[4] W. Ma and D. H. C. Du. 2004. Design a progressive video caching policy for video proxy servers. IEEE Trans. Multimed. 6(4): 599-610.

[5] V. Pacifici, F. Lehrieder, and G. Dán. 2016. Cache Bandwidth Allocation for P2P File-Sharing Systems to Minimize Inter-ISP Traffic. IEEEACM Trans. Netw. 24(1): 437-448.

[6] K. M. Wilson and K. Olukotun. 2001. High bandwidth on-chip cache design. IEEE Trans. Comput. 50(4): 292-307.

[7] Q. Zhang, Z. Xiang, W. Zhu, and L. Gao. 2004. Cost-based cache replacement and server selection for multimedia proxy across wireless Internet. IEEE Trans. Multimed. 6(4): 587-598.

[8] Janaki K, Indhumathi K, Vijayakumar P, and Ashok Kumar K. 2015. A novel approach for a high performance lossless cache compression algorithm. ARPN J Eng Appl Sci ARPN J. Eng. Appl. Sci. 10(7): 3178-3184.

[9] Tanwir, G. Hendrantoro, and A. Affandi. 2015. Early result from adaptive combination of LRU, LFU and FIFO to improve cache server performance in telecommunication network.Presented at the Intelligent Technology and Its Applications (ISITIA), 2015 International Seminar on. pp. 429-432.

[10] F. Hameed, L. Bauer, and J. Henkel. 2013. Simultaneously optimizing DRAM cache hit latency and miss rate via novel set mapping policies. in 2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES). pp. 1-10.

[11] M. Akon, M. T. Islam, X. Shen, and A. Singh. 2010. Hit Optimal Cache for Wireless Data Access. in 2010

IEEE Global Telecommunications Conference (GLOBECOM 2010). pp. 1-5.

[12] M. Hashemi, Khubaib, E. Ebrahimi, O. Mutlu, and Y. N. Patt. 2016. Accelerating Dependent Cache Misses with an Enhanced Memory Controller. in 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). pp. 444-455.

[13] V. Venkatesan, Y. C. Tay, Y. I. Zhang, and Q. Wei. 204. A 3-Level Cache Miss Model for a Nonvolatile Extension to Transcendent Memory.Presented at the Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on. pp. 218-225.

[14] J. van den Berg and D. Toswley. 1999. Properties of the miss ratio for a2-level storage model with LRU or FIFO replacement strategy and independent references. IEEE Trans. Comput. 42(4): 508-512.

[15] Y. Zhong, S. G. Dropsho, X. Shen, A. Studer and C. Ding. 2007. Miss Rate Prediction across Program Inputs and Cache Configurations. IEEE Trans. Comput. 56(3): 328-343.

[16] M. Lenjani and M. R. Hashemi. 2014. Tree-based scheme for reducing shared cache miss rate leveraging regional, statistical and temporal similarities. IET Comput. Digit. Tech. 8(1): 30-48.

[17] S. Wan, Q. Cao, X. He, C. Xie and C. Wu. 2008. An Adaptive Cache Management Using Dual LRU Stacks to Improve Buffer Cache Performance.Presented at the 2008 IEEE International Performance, Computing and Communications Conference. pp. 43-50.

[18] J. Shim, P. Scheuermann, and R. Vingralek. 1999. Proxy cache algorithms: design, implementation, and performance. IEEE Trans. Knowl. Data Eng. 11(4): 549-562.