ISSN: 0975-833X

Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 9, Issue, 01, pp.45632-45638, January, 2017

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

RESEARCH ARTICLE

NEW AVERAGE LEAST FREQUENCY USED WEB CACHE REPLACEMENT USING INTELLIGENT AGENT VS LFU, LRU, SIZE AND PCCIA CACHE REPLACEMENT TECHNIQUES FOR SAMPLE GENERATED WITH WEBTRAFF

***,[1]Mohammed Salah Abdalaziz Khaleel, [2]Saif Eldin Fattoh Osman and [2]Hiba Ali Nasir Sirour**

[1]Faculty of Computer Studies, International University of Africa, Khartoum, Sudan
[2]Emirates College of Technology, Faculty of Computer Studies IUA, Khartoum, Sudan

ARTICLE INFO

Article History:
Received 16th October, 2016
Received in revised form
20th November, 2016
Accepted 25th December, 2016
Published online 31st Januarv, 2017

Key words:

LFU, Proxy cache, Removablepolicies,
Web caching performance.

ABSTRACT

Web cache replacement plays important role in increasing the performance and speed of browsing web sites using internet. This paper highlights a new proposed Average Least Frequency Used Removal (ALFUR) and compares it with web cache replacement techniques like (LFU, LRU, SIZE, and PCCIA). Hit Ratio (HR) and ByteHit Ratio (BHR) were used to measure the performance of these algorithms, and it was found that ALFUR technique has the bestHit Ratio and ByteHit Ratio since it has the highest values for HR &BHR when cache size was started from 1Mb,6Mb,500Mb,800Mb.

Citation: Mohammed Salah Abdalaziz Khaleel, Saif Eldin Fattoh Osman and Hiba Ali Nasir Sirour, 2017. "New average least frequency used web cache replacement using intelligent agent VS LFU, LRU, size and Pccia cache replacement techniques for sample generated with Webtraff", *International Journal of Current Research*, 9, (01), 45632-45638.

INTRODUCTION

Consider a caching network proxy application for the HTTP protocol. This proxy typically sits between the internet and the user or a set of users. It ensures that all the users are able to access the internet and enables sharing of all the shareable resources for optimum network utilization and improved responsiveness. Such a caching proxy should try to maximize the amount of data that it can cache in the limited amount of storage or memory that it has at its disposal (Prof. Ketan Shah Anirban Mitra Dhruv Matani, 2010). Typically, lots of static resources such as images, CSS style sheets and JavaScript code can very easily be cached for a fairly long time before it is replaced by newer versions. These static resources or "assets" as programmers call them are included in pretty much every page, so it is most beneficial to cache them since pretty much every request is going to require them. Furthermore, since a network proxy is required to serve thousands of requests per second, the overhead needed to do so should be kept to a minimum.To overcome this situation, Web caching technique has been used. Web cache reduces the high traffic over the internet so that user can access the web content faster.

****Corresponding author: Mohammed Salah Abdalaziz Khaleel,***
Faculty of Computer Studies, International University of Africa, Khartoum, Sudan.

The main purpose of cache is to place the copy of object near to theclient, so that web user can access the object easily, without the request going to the web server. That keep the need object near as demand to decrease the overhead of network to load the missing object form proxy server or original server and increase the availability of object. There are different points where a cache can be set up, such as browser, proxy server and close to server. When a user requests a web page, firstly it is checked in cache, if the requested web page is available then it send back to the user. If the web page is not found in the cache; then the request is redirected to the web server and sends the response to the client. Because of the limited size of memory of cache, it becomes much hard to save all objects in the memory (Prof. Ketan Shah Anirban Mitra Dhruv Matani, 2010).Traditional replacement policies but not efficient in web caching are still used by most web browsers (Wessels,2001; Tan*et al*., 2006). In fact, a replacement policy can be effected by few important factors of web objects (Chung-yi Chang*et al*., 2010; Tan*et al*., 2006; Koskela*et al*., 2003). These factors include but not limited to recency (i.e., time of the last reference to the object), frequency (i.e., number of the previous requests to the object), size, and access latency of the web object. These factors can be combined into the replacement decision. Most of the proposed approaches in the literature use one or more of these factors without paying attention of combining some of these factors. However, combination of

these factors is still a challenging task as one factor in a particular.This paper highlights ALFUR and compares it with LFU, LRU, SIZE and fuzzy logic base policy.

LFU ReplacementStrategy

Since a network proxy is required to work on thousands of requests per second, the overhead needed to do so should be kept to a least possible. To minimize this overhead, the network proxy should force out resources that are not frequently used. Hence, only the frequently used resources should be kept at the expense of the not frequently used ones since the former have verified themselves to be valuable over a period of time. Static resources of heavily used pages are always requested by every user of that page. LFU cache replacement strategy is one of the Web caching techniques that can be activated by these caching proxies to force out the least frequently used items in its cache (Cherkasova *et al*., 2001). The main characteristics of this technique is that LFU is keeping track of the number of times a block is referenced in memory, and when more spaces is needed for new objects but there is no more rooms in the cache because it is full; in such case the system will remove the item with the lowest reference frequency from the cache. The major disadvantage of the LFU replacement algorithm is that some web sites possess their place in cache memory for a long time even without using them again. This leads to wasting a certain size of cache memory since this element remains in the memory with no change. Other disadvantages of LFU policies are that they involve logarithmic implementation complexity in cache size, and they almost pay no attention to recent history.

Weighting-Replacement-Policy (WRP)

In order to enhance the performance of the LFU algorithm, a replacement based on Weighting-Replacement-Policy (WRP) was proposed (Amany Sarhan *et al*., 2015). This algorithm acts like LFU by exchanging pages that were not recently used and pages that are called up only once. Three counters are used in ranking the pages in cache memory; the counter which shows the recency of block (L), the counter which shows the number of times that block buffer has been referenced (F), and the time difference (ΔT) between the last access time (Tc) and time of penultimate (Tp). Thus, the weighting value of block i can be calculated by the following equation: $Wi=Li/(Fi*\Delta Ti)$. The time between each reference to a block would be at least one in its lowest case. In every access to buffer, if referenced block j is in the buffer then a hit is occurred and this policy will work as follows: - Li will be changed to Li+1for every $i \neq j$. - For i = j first we put $\Delta Ti = Li$, $Fj = Fj+1$ and then $Lj=0$ But if referenced block j is not in the buffer, a miss occurs and the algorithm will choose the block in buffer which its weighting function value is greater than the others. This will be done from top to down.

In this way, if values of some object are equal to each other, the object which has upper place in the buffer will be chosen to be forced out from buffer. It means that WRP policy follows FIFO law in its nature. Let assume that a miss has been occurred and block k has the greatest weighting value and then it should be forced out from buffer. First we change Li to Li + 1 for every $i \neq k$ and then replace new referenced block with block k. The final step is to s*et al*l weighting factors of block k to their initial values. The weighting value of the blocks that are in buffer will be updated in every access to cache.

Frequency based & Recency-based strategies

Most of these strategies are an extension of the commonly known algorithm Least Frequently Used (LFU). There are two methods to implement these algorithms, one requires the use of supporting cache, and the others are not. Spatial Locality is a property of request streams concerns with the probability that an object will appear again based on how often it's been seen before. This property is used by Frequency based strategies, contrasting Recency-based strategies, these algorithms require complex data structures, such as binary heaps to help decrease the time overhead in making their decisions.

Comparatively, most recency-based strategies only requires to keep track of the most recent values seen by the proxy cache, simplifying the record of a web object's data to the time it is in the cache even if it is removed and added repetitively. However, frequency counts do not concern only to the lifespan of a certain object in the cache, but can also be determined across several lifetimes of the object. The persistent recording of data for an object's frequency counts is known as Perfect LFU, which definitely needs more space overhead. The tracking of data while the object is only in the cache is known as In-Cache LFU. Since there is space overhead with perfect LFU, the in-cache is concerned as one of these strategies (Sam Romano and Hala ElAarag, 2008).

LRU cache replacement

Traditional caching policies are suitable for CPU caches and virtual memory systems. Although most Web proxy servers still concern with these conventional policies they are not efficient in Web caching area. Least-Recently Used (LRU) algorithm is the simplest and most common cache management approach, which removes the least recently accessed objects until there is enough space for new objects. LRU is easy to implement and proficient for identical size objects, like in the memory cache. However, it does not work well in Web caching since it does not consider the size or the download latency of objects, see Table (2) for comparison (Waleed Ali *et al*., 2011).

SIZE cache replacement

The SIZE policy is one of the common web caching approaches. When this algorithm is used and space is needed for a new object, it replaces object(s) with the largest size from cache by new one. The main disadvantage of SIZE policy is that objects with small size are remaining in the cache even though they are not accessed again, this leads to cache pollution. Greedy-Dual-Size (GDS) policy was suggested as extension of the SIZE policy to clean up the cache pollution. This algorithm integrates several factors and allocates a key value or priority for each web object stored in the cache. When cache space becomes unavailable and new object is required to be stored in cache, the object with the lowest priority is removed (Waleed Ali and Siti Mariyam Shamsuddin, 2015). Although, these conventional Web caching approaches suffer from some limitations they form the basis of other efficient caching algorithms and most Web proxy servers are still concern with these mentioned earlier conventional replacement policies.

Fuzzy logic base replacement policy

PCCIA is Fuzzy logic base algorithms, is implemented on two sides; parent cache side and child caches side and it was developed from combining LFU, LRU and Size caching replacement policies with a Cache Cleaner Agents use fuzzy logic to make an intelligent decision. In this algorithm LFU and LRU policies are performed on the child caches side. And Cache Cleaner Agents are responsible of examining objects' key values and remove web object with high clean up priority proactively. To complete the cleanup task in proficient manner reactive Coordination has been applied between the parent and child cleaner agents to increase hit ratio and byte hit ratio; their common goal.

When a web object with medium priority is encountered in parent and children caches, coordination rules are applied by the Coordination agent. In similar state the cleaner agent applies Q-learning algorithm to avoid difficult calculation to take a suitable action. Q-learning algorithm associates reward value to each action. Optimal action that leads to the goal, has an instant high reward while other actions have low reward values. A graph can be used to represent States and Actions, node represent "state" while agent's movement from one node to another represent the "action". Web traff simulator that generates two samples of workload was used for testing purposes. These samples represented the users' requests and used cache sizes. To evaluate the cache performance, Hit Ratio and Byte Hit Ratio were used. Simulation results show that when the cache size growth PCCIA performs better than LRU, LFU and Size replacement polices in terms of hit rate and byte hit rate (Hiba. A. Nasir *et al*., 2013).

ALFUR Description

ALFURis a new multi agent Technique that consists of four big agents: Reader agent, Analyzer agent, Removal agent and Performance agent. JADE technology was used to implement these agents, while Java Programming Language was used to write codes for these agents with their tasks.

These agents are discussed as following:

Reader Agents: The Reader agent reads the object date from "access log file" which is created by the proxy server.

Analyzer Agents: The main task of the Analyzer Agents is the calculations of frequency, size and request time for objects as needed to prepare object's information then send it to removal agent.

Removal Agents: Its main task is to remove the object to create free space in cache for other object, depending on the Analyzer Agent results.

Performance evaluator Agent: calculate the number of hit ratio and number of byte hit ratio that are used to measure the performance of ALFUR.

Model Architecture

The model architecture consists of three modules as shown in Figure1:

1-The Monitoring Module: This contains the monitoring agent and Analyzer Agent. It is a reactive agent that monitors the proxy cache. This agent works by using a fast response behavior. It provides information that allows the Analyzer Agent agents to take a decision.

2- The Removal Module: This contains the parent cache Removal agents ; which task is to clean up the cache according to web object frequencies, sizes, and times.

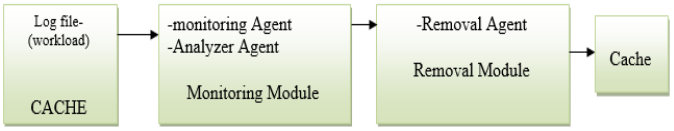

Figure 1. Model Architecture

Algorithm Methodology

The new proposed Technique-*ALFUR*- reads the objects in cache memory using reader agent then it analyzes the cache object using analyzer agent that calculates average of frequencies, object's frequency, size and time. The average frequency calculating the summation of total number of frequency over number of objects, while the average size is calculating the summation of object size over total object size. The removal is remove based on the following conditions:

- If object's frequency greater than the average of frequencies then don't remove this object, either that removes this object.
- If object's frequency equals to the average of frequencies then compare the object's size with the average, if it's greater it must be removed, either that it's not removed.
- If object's frequency equals to the average frequencies and the average equals to the object's size, then it calculates the average of the web objects' time stamp if it less than web object time stamp not remove, either that this object must be removed.
- Finally, the performance agent calculates the number of hit ratio and number of byte hit ration to measure the performance of this new Technique.

ALFUR Algorithm: New LFU Caching Algorithm

1. Read web object
2. Calculate web object frequency, size ,request time
3. Calculate Average of web object frequency, size and request time
4. IF object Freq>Average objects Freaq THEN
5. ((Don't remove object
6. else
7. Remove object)
8. Else IF object Freq=Average objects and object size>average objects size THEN
9. (Remove object
10. else
11. Don't remove object)
12. Else IF object Freq=Average object and object size=average object size and object request time>average request time THEN
13. (Don't remove object
14. Else

15. remove object))
16. end

Performance Metrics

To evaluate the performance of the cache, performance metrics are being used. These performance metrics play a very important role in the web cache performance calculation. Based on these performance metrics we can compare the performance of different algorithms. Cache replacement policy depends on the several key metrics. The most commonly used are Hit rate, Byte hit rate.

Hit rate

The percentage of all requested objects which are found in the cache instead of transferred from the requested server.

Byte hit rate

The percentage of all data that is transfer straight from the Cache rather than from requested server. Table 1 shows how they can be calculated.

Table 1. Performance metrics (Amany Sarhan *et al*., 2015)

Metric	Definition
Hit Ratio	
Byte Hit Ratio	
When n: total Number of requests ∂i: 1 if the request i is in the cache ∂i: 0 otherwise bi: size in bytes	

Table 2. Comparison between standard replacements (Waleed Ali *et al*., 2011)

Policy	Brief description	Advantages	Disadvantages
LRU	The least recently used objects are removed first	Simple and efficient with uniform size objects, such as the memory cache	Ignores download latency and the size of Web objects
LFU	The least frequently used objects are removed first	Simplicity	Ignores download latency and size of objects and may store obsolete Web objects indefinitely
SIZE	Big objects are removed first	Prefers keeping small Web objects in the cache, causing high cache hit ratio	Stores small Web objects even if these object are never accessed again. Low byte hit ratio

RESULTS

The following figures give results of ALFUR with traditional LFU, LRU, Size and PCCIA removable policies in terms of Hit Ratio and Byte Hit rate

Hit Ration comparison between ALFUR and other Replacements Removal Algorithms

The following figures show the comparison of ALFUR with traditional LFU, LRU, SIZE and PCCIA removable algorithms in terms of Hit ratio. For different cache size shows above the figures.

Hit ratio Comparison Figure 2 Tested with cache size 1MB

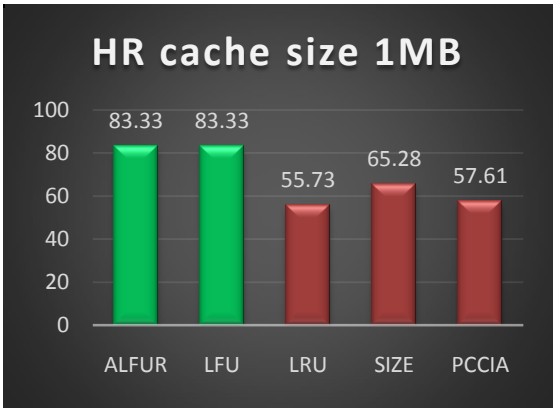

Figure 2. HR when cache size=1MB

Hit ratio Comparison Figure 3 Tested with cache size 6MB

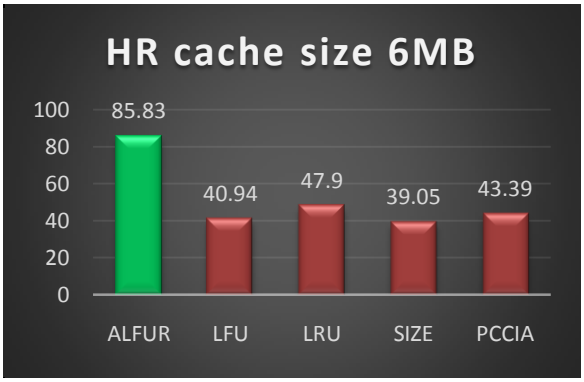

Figure 3. HR when cache size=6MB

Hit ratio Comparison Figure 4 Tested with cache size 500MB

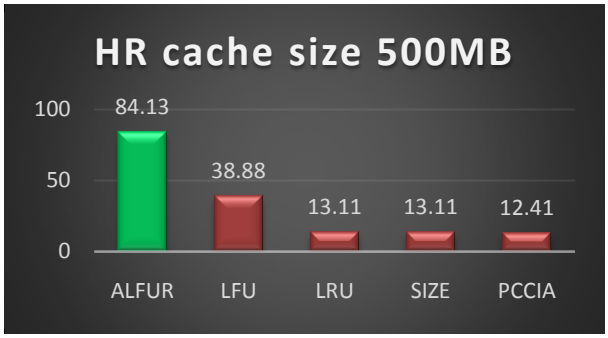

Figure 4. HR when cache size=500MB

Hit ratio Comparison Figure 5 Tested with cache size 800MB

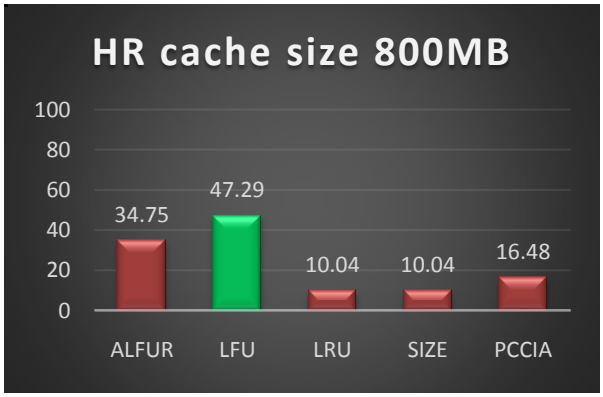

Figure 5. HR when cache size=800MB

Byte Hit Ration comparison between ALFUR and other Replacements Removal Algorithms

The following figures show the comparison of ALFUR with traditional LFU,LRU, SIZE and PCCIA removable algorithms in terms of Byte Hit ratio. For different cache size shows above the figures.

Hit ratio Comparison Figure 6 Tested with cache size 1GB

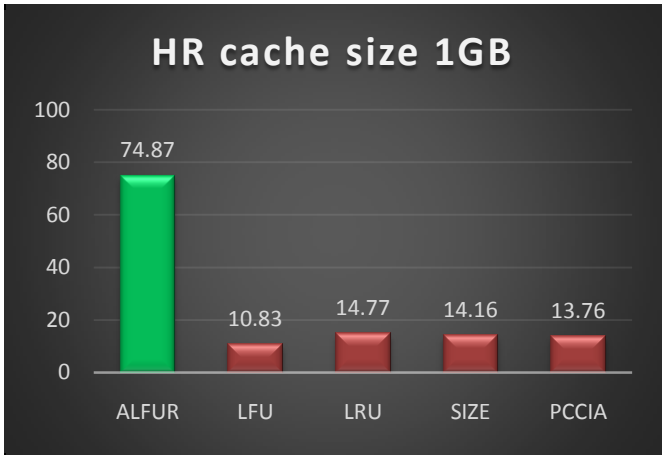

Figure 6. HR when cache size=1GB

Byte Hit ratio Comparison Figure 7 Tested with cache size 1MB

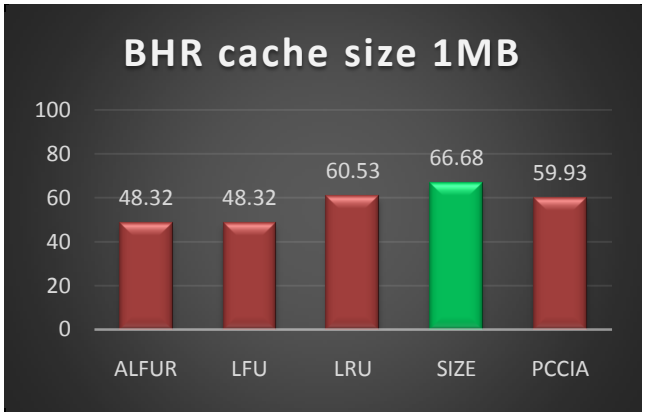

Figure 7. BHR when cache size=1MB

Byte Hit ratio Comparison Figure 8 Tested with cache size 6MB

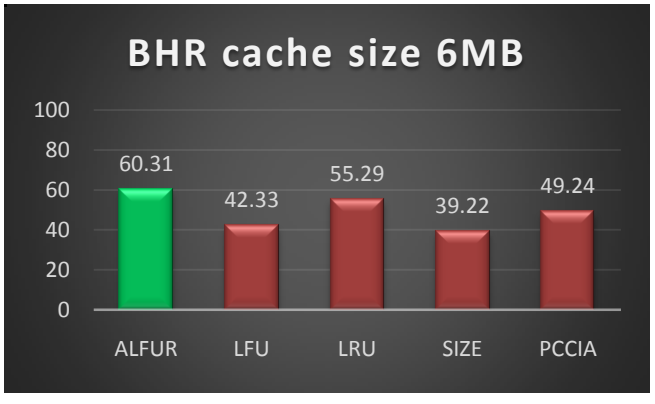

Figure 8. BHR when cache size=6MB

Byte Hit ratio Comparison Figure 9 Tested with cache size 500MB

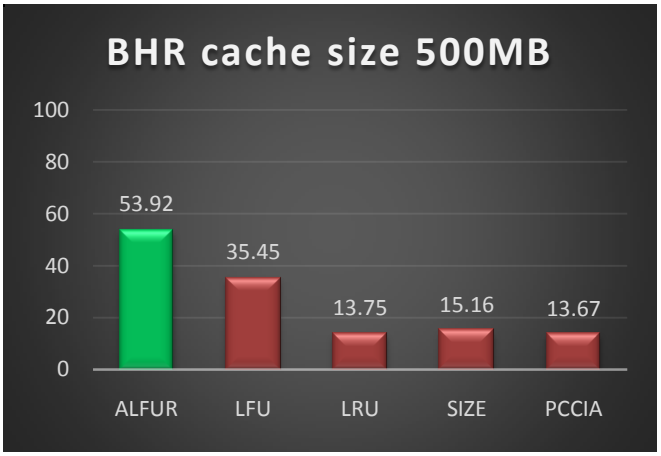

Figure 9. BHR when cache size=500MB

Byte Hit ratio Comparison Figure 10 Tested with cache size 800MB

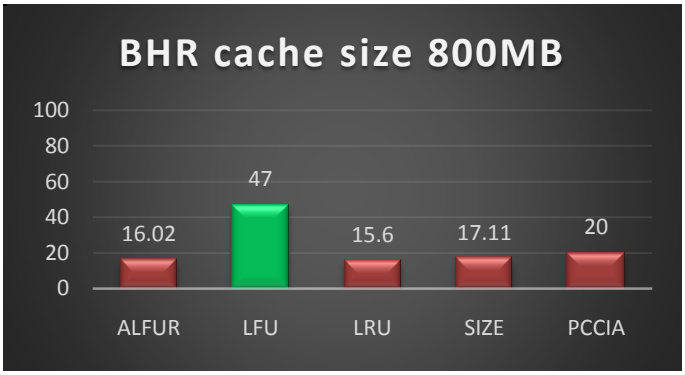

Figure 10. BHR when cache size=800MB

Byte Hit ratio Comparison Figure 11 Tested with cache size 1GB

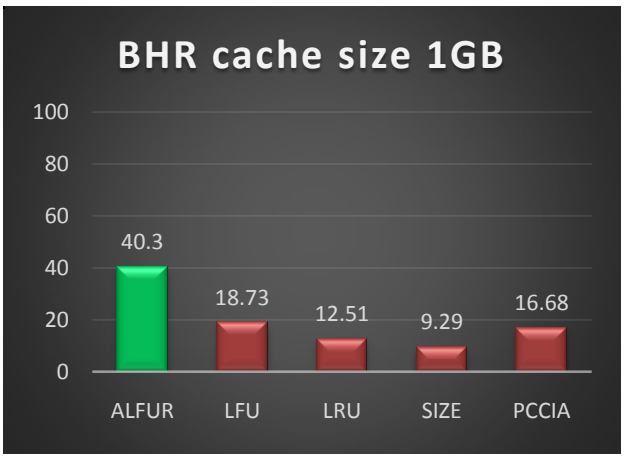

Figure 11. BHR when cache size=1GB

ALFUR Best Result In Hit Ratio and Byte Hit Ratio

The Figure12 shows the best result of ALFUR in term of Hit Ration and Byte Hit Ration for generated data using webtraff simulator, when cache size equals 6MB.

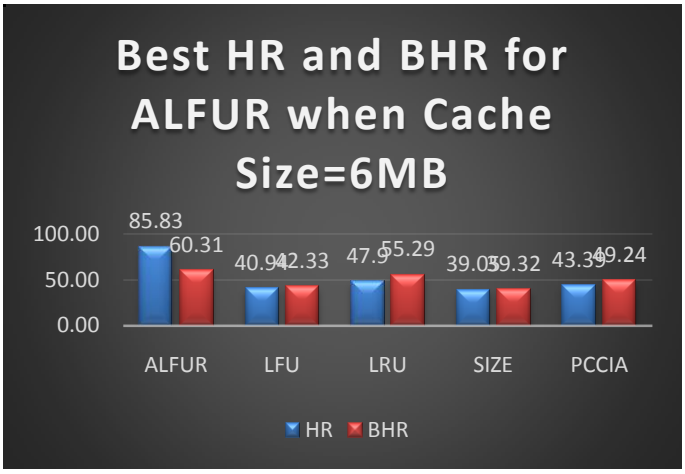

Figure 12. Best HR and BHR for ALFUR

ALFUR Worst Result In Hit Ratio and Byte Hit Ratio

The Figure13 shows the best result of ALFUR in term of Hit Ration and Byte Hit Ration for generated data using webtraff simulator, when cache size equals 800MB

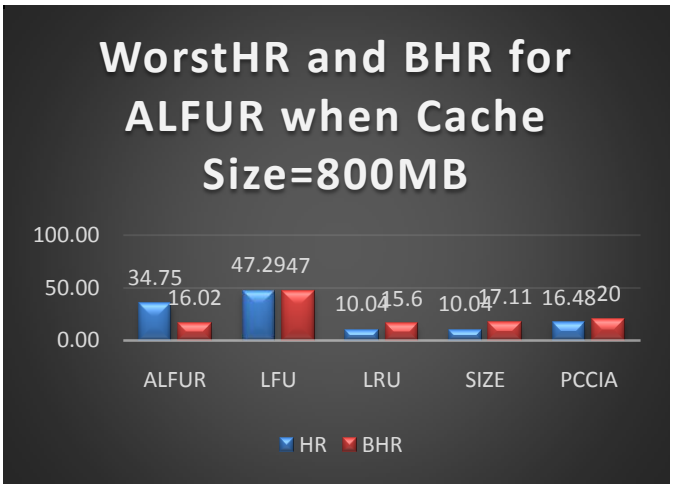

Figure 13. Worst HR and BHR for ALFUR

RESULTS AND DISCUSSION

Result Discussion in terms of Hit Ratio for all Replacement Algorithms

This section consists of performance result discussion for data generated using webtraff in the term of Hit Ratio between all replacement algorithms (ALFUR, LFU, LRU, SIZE, and PCCIA).

- Figure 2, the arrange of the replacement algorithms depends on cache size 1Mb, the ALFUR and LFU are same best result in this sample with hit ratio 83.33%, the replacement algorithm is in the second range is SIZE with 65.28% .the replacement algorithm is in the third range is PCCIA with 57.61%the replacement algorithm is in the last range is LRU with 55.73%HR.ALFUR is better than second range in this sample SIZE with rate +18.05%, this rate of hit ratio for ALFUR is forth range for all samples.
- Figure 3, the arrange of replacement algorithms depends on cache size 6Mb, the ALFUR is in the first range with 85.83% HR, LRU is in the second range with 47.9% HR, PCCIA in the third range with 43.39% HR, LFU is in the fourth range with 40.94% HR and SIZE replacement in the last range with 39.05% HR. ALFUR is better than second range in this sample LRU with rate +37.93%. This rate of hit ratio for ALFUR is third range for all samples.
- Figure 4, the arrange of replacement algorithms depends on cache size 500Mb, the ALFUR is in the first range with 84.13% HR, LFU is in the second range with 38.88% HR,LRU and SIZE in the third range with 13.11% HR and PCCIA is in the last range with 12.41% HR . ALFUR is better than second range in this sample LFU with rate +45.25%. This rate of hit ratio for ALFUR is second range for all samples.
- Figure 5, the arrange of policies depend on cache size 800Mb , the LFU is in the first range with 47.29% HR, ALFUR is in the second range with 34.75% HR, PCCIA is in the third range with 16.48% HR .LRU and SIZE in the last range with 10.04% HR . ALFUR is worse than first range in this sample LFU with rate -12.54%. This rate of hit ratio for ALFUR is fifth range for all samples.
- Figure 6, the arrange of policies depend on cache size 1Gb, the ALFUR is in the first range with 74.87% HR, LRU is in the second range with 14.77% HR, SIZE is in the third range with 14.16% HR, PCCIA in the fourth range with 13.76% HR and LFU in in the last range with 10.83% HR. ALFUR is better than second range in this sample LRU with rate +60.1%. This rate of hit ratio for ALFUR is first range for all samples.

Result Discussion in the term of Byte Hit Ratio for all Replacement algorithms

This section consists replacement algorithms of result discussion performance for data generated using webtraff in the term of Byte Hit Ratio between all replacement algorithms (ALFUR, LFU, LRU, SIZE and PCCIA).

- Figure 7, the arrange of replacement algorithms depends on cache size 1Mb ,the SIZE is in the first range with 66.68% BHR, LRU is in the second range with 60.53% BHR, PCCIA is in the third range with 59.93% BHR .ALFUR and LFU in the last range with 48.32% BHR . ALFUR is worse than first range in this sample SIZE with rate -18.36%. This rate of byte hit ratio for ALFUR is forth range for all samples.
- Figure 8, the arrange of replacement algorithms depends on cache size 6Mb, the ALFUR is in the first range with 60.31% BHR, LRU is in the second range with 55.29% BHR, PCCIA is in the third range with 49.24% BHR .LFU in the fourth range with 42.33% BHR and SIZE in the last range with 39.22% BHR. ALFUR is better than second range in this sample LRU with rate +5.02%. This rate of byte hit ratio for ALFUR in third range for all samples.
- Figure 9, the arrange of replacement algorithms depends on cache size 500Mb, the ALFUR is in the first range with 53.92% BHR, LFU is in the second range with 35.45% BHR, SIZE is in the third range with 15.16% BHR .LRU in the fourth range with 13.75% BHR and PCCIA in the last range with 13.67% BHR. ALFUR is better than second range in this sample LFU with rate +18.47%. This rate of byte hit ratio for ALFUR is second range for all samples.

- Figure 10, the arrange of replacement algorithms depends on cache size 800Mb, the LFU is in the first range with 47% BHR, PCCIA is in the second range with 20% BHR, SIZE is in the third range with 17.11% BHR .ALFUR in the fourth range with 16.02% BHR and LRU in the last range with 15.60% BHR. ALFUR is worse than first range in this sample LFU with rate -30.98%. This rate of byte hit ratio for ALFUR is fifth range for all samples.
- Figure11, the arrange of replacement algorithms depends on cache size 1Gb, the ALFUR is in the first range with 40.3% BHR, LFU is in the second range with 18.73% BHR, PCCIA is in the third range with 16.68% BHR .LRU in the fourth range with 12.51% BHR and SIZE in the last range with 9.29% BHR. ALFUR is better than second range in this sample LFU with rate +21.57%. This rate of byte hit ratio for ALFUR is first range for all samples.

ALFUR Best Result Discussion (HR and BHR)

Generally the hit rate increases when cache size increases, the best HR rate and BHR rate for ALFUR in all sample when cache size is equal to 6MB that is shown in 12.The best Hit Ratio equals 85.83% and the best Byte Hit Ratio equal 60.31%.

ALFUR Worst Result Discussion (HR and BHR)

The worst HR rate and BHR rate for ALFUR in all sample is when cache size is equal to 800MB that is shown in figure 13. The reason for the worst ALFUR case in this result is because the average frequency for this sample is very big when comparing with other web object frequency have very small number of frequency. The worst Hit Ratio equals 34.75% and the worst Byte Hit Ratio equal 16.02%.

Conclusion

Although many web caching policies have been proposed in the literature, they still have lots of overheads and are difficult to implement. In this paper, a new replacement policy is developed in order to overcome some of the problems found in the literature. The proposed strategy was able to evict the object with small frequency, size and oldest web object in cache. This was seen in the simulation results through calculating the hit ratio and Byte Hit Ratio. The simulation results showed that proposed. Best HR Result ALFUR for all Cache Replacements (ALFUR, LFU, LRU, SIZE, PCCIA) in terms of hit ratio form all generated samples, From above result the average Hit Ratio for ALFUR is the best performance with 72.58% HR, second is LFU with 44.25% HR, third PCCIA with 28.73% HR, forth SIZE with 28.33% HR and last range is LRU with 28.31% HR. Best BHR Result ALFUR for all Cache Replacements (ALFUR, LFU, LRU, SIZE, PCCIA) in terms of byte hit ratio form all generated samples, From above result the average Byte Hit Ratio for ALFUR is the best performance with 43.77% BHR, second is LFU with 38.37% BHR, third PCCIA with 31.90% BHR, forth range is LRU with 31.54% BHR and the last range is SIZE with 29.51% BHR. From above result ALFUR is better than LFU, LRU, SIZE, and PCCIA.

REFERENCES

Amany Sarhan, Ahmed M. Elmogy, Sally Mohamed Ali ,"A new Web Cache Replacement Approach based on Internal Requests factor", *IJCSNS International Journal of Computer Science and Network Security*, VOL.15 No.3, March 2015

Cherkasova, Ludmila, and Gianfranco Ciardo. 2001. "Role of aging, frequency, and size in web cache replacement policies." In High-Performance Computing and Networking, pp. 114-123. Springer Berlin Heidelberg.

Chung-yi Chang,Tony McGregor,Geoffrey Holmes, 2010. "The LRU* WWW proxy cache document replacement algorithm"Sep 17.

Hiba. A. Nasir, Yahia. A. Mohammed, Amir. A. Eisa, 2013. "Agent-based Proxy Cache Cleanup Model using Fuzzy Logic", proceedings of International Conference of Computing Electrical and Electronic Engineering (ICCEEE), Khartoum, Sudan, August.

Koskela, T., J.Heikkonen, and K.Kaski. 2003. "Web Cache Optimization with Nonlinear Model Using Object Feature," Computer Networks Journal, Elsevier, 20 Dec.

Prof. Ketan Shah Anirban Mitra Dhruv Matani,An O(1) algorithm for implementing the LFU cache eviction scheme,August 16, 2010.

Sam Romano and Hala ElAarag, 2008. "A Quantitative Study of Recency and Frequency based Web Cache Replacement Strategies", ISBN 1-56555-318-7.

Tan, Y., Y. Ji, and V.S Mookerjee. 2006. "Analyzing DocumentDuplication Effects on Policies for Browser and Proxy Caching". *INFORMS Journal on Computing*, 18(4), 506522.

Waleed Ali andSitiMariyamShamsuddin, 2009. "Integration of Least Recently Used Algorithm and Neuro-FuzzySystem into Client-side Web Caching," *International Journal of Computer Science and Security (IJCSS)*.

Waleed Ali, Siti Mariyam Shamsuddin, 2015. "Intelligent Dynamic Aging Approaches in Web Proxy Cache Replacement, *Journal of Intelligent Learning Systems and Applications*, 7, 117-127

Waleed Ali, Siti Mariyam Shamsuddin, and Abdul Samad Ismail, 2011. "A Survey of Web Caching and Prefetching", Int. J. Advance. Soft Comput. Appl., Vol. 3, No. 1, March 2011 ISSN 2074-8523; Copyright © ICSRS Publication.

Wessels, Duane. Web caching. O'Reilly Media, Inc., 2001.

Wong. A.K.Y. 2006. "Web Cache Replacement Policies: A Pragmatic Approach," IEEE Network,Magazine.

****