ijar
ISSN NO. 2320-5407

Journal Homepage: -www.journalijar.com
INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI:10.21474/IJAR01/3048
DOI URL: http://dx.doi.org/10.21474/IJAR01/3048

RESEARCH ARTICLE

IMPROVEMENT OF LEAST FREQUENCY USEDWEB CACHE REPLACEMENT TECHNOLOGY USING INTELLIGENT AGENTS.

Mohammed Salah Abdalaziz Khaleel[1], Saif Eldin Fattoh Osman[2] and Hiba Ali Nasir Sirour[2].
1. Faculty of Computer Studies, International University of Africa, Khartoum, Sudan.
2. Emirates College of Technology, Faculty of Computer, Studies IUA, Khartoum,Sudan.

Manuscript Info

Manuscript History

Received: 30 November 2016
Final Accepted: 28 December 2016
Published: January 2017

Key words:-
LFU; proxy cache; removable policies; Web caching performance.

Abstract

The rapid development of the Internet needs to meet development and improvement of techniques to increase the performance of browsing and decrease the overload on the network traffic To overcome this situation, Web caching techniques has been used. Web cache reduces the high traffic over the Internet so that users can access the web content faster.
The ALFUR replacement algorithm is developed by taking advantage of LFU, LRU and SIZE cache replacement algorithms. This algorithm separates the operation of the algorithm into three main agents, that we use it because to have the benefit of agents. This paper is introduce the new proposed technique and compare between traditional LFU and ALFUR.

Introduction:-

Nowadays, the rapid development of computer networks and internet services, the interactive web pages especially the social media have been accompanied by a mass and an extension in the speed of browsing and access to information at high speed. This speed is the causative agent that is dealing with the cache memory on browser side, proxy or origin server side. In order to be in a high speed browsing, should be dealing mainly with the cache memory of web caching concepts, since these pages must be carried into cache memory. While the main issues lie in the cache memory on any side have some limitations of size, it might be possible to the main memory to be fully occupied depending on the data or object that is stored in it. So, looking at this scenario in a serious way at this important junction, the proposed web caching policy of this study tends to tackle this issues immediately as it is continuously happening.

The sample of data is generated using webtraff generator, to compare between LFU and ALFUR the researcher generated 5 sample of data with cache size 1Mb, 6Mb, 500Mb, 800Mb and1Gb.

There are different points at which we can set up a cache such as browser at client site, proxy server and close to original server. When a user requests web page, firstly it is checked in cache, if the requested web page is available then is sent back to the user. If the web page is not found in the cache, then the request is redirected to the web server and the response is sent to the client. Because of the limited size of the cache memory, it becomes so hard to save all objects in this memory [1].

Corresponding Author:-Mohammed Salah Abdalaziz Khaleel.
Address:-Faculty of Computer Studies, International University of Africa, Khartoum,Sudan.

The new proposed algorithm to improve LFU web cache replacement is named ALFUR - the proposed algorithm - combines between main web caching replacement strategies: LFU, Least Redundancy Used (LRU) and SIZE respectively; to take advantage of their features.

ALFUR is designed to solve the problem of the traditional web caching replacement strategies; to decrease the overhead of network traffic and to increase the speed of browsing. This paper is implemented using artificial intelligent that is play main role to improve algorithms.

LFU replacement strategy:-
LFU replacement strategy is one of web cache replacement strategy. That used to remove object from cache memory to free space for new objects. The cache memory play main role with increase the speed of browsing throw internet. Mainly LFU depends on frequency number, Frequency number is increased once when web object is requested. The object with least frequency is removed from the cache [1].

Disadvantage of LFU:-
Since LFU is a frequency-based strategy, it considers the frequency of access. This is valuable in static environments where the status of objects does not change very much over a specific time period (day, week).

The disadvantages are as follows [1]:-
Complexity. LFU-based strategies involve a more complex cache administration. LFU cannot be implemented, for example without a priority queue.

Cache pollution. Frequency counts are static for dynamic changes in the Workload. Therefore, aging was introduced. But aging is nothing but a recency- based technique. It is questionable if sophisticated aging techniques are better than simple recency-based techniques in dynamically changing environments. Furthermore, they add complexity to the replacement process.

Similar values. Many objects can have the equivalent frequency count. In this case, a tie breaker factor is required.

Category of cache page Replacement policy:-
Cache page replacement policies can be divided into four categories as follows:

Recency-based:-
In this category, time is the main issue, i.e. time to access the last references of object. An example of this category is LRU (Least Recently Used) that has been applied in a number of proxy caching servers.

Size-based:-
In this category, object size is the basic parameter. LFU-Size based algorithm is considered an example of this category.

Frequency-based:-
In this category, the frequency of the object is functioned, that number of times an object is accessed is worked on. An algorithm of this category is LFU.

Function-based:-
This category involves multiple parameters which are related to performance metric that was used to determine a cost based function algorithms. Most suggestive algorithm of this category is Greedy-Dual Size. [3], [4], [7].

Agent:-
There is no commonly recognized definition of the word agent. Russel and Norvig (1995) define an agent as an entity that can be viewed as perceiving its environment through sensors and acting upon its environment through effectors.

(Coen, 1995) views software agents as programs that engage in dialogs and negotiate and coordinate the transfer of information.

Wooldridge and Jennings (1995) state that an agent is a hardware and/or software-based computer system displaying the properties of autonomy, social adeptness, reactivity, and proactivity. Others (Brustolini, 1991; Franklin and Graeser, 1996; Maes, 1995; Hayes-Roth et al, 1995; Gilbert et al, 1995) offer variants on this theme.

There is an agreement with Autonomy; the ability to act without the involvement of humans or other systems, is a significant feature of an agent. Furthermore, different attributes take on different importance based on the domain of the agent. [5]

Intelligent agent:-
Wooldridge and Jennings (1995) define an intelligent agent as one that is capable of flexible autonomous action to meet its design objectives.

Flexible means:-
Reactivity:-
Intelligent agents identify and respond in a timely fashion to changes that arise in their environment in order to fulfill their design objectives. The agent's aims and/or rules that form the root for a process that is currently executing may be affected by a changed environment and a different set of actions might be needed to be accomplished.

Pro-activeness:-
In a changed environment, intelligent agents have to recognize opportunities and take the initiative if they are to produce meaningful results. The challenge to the agent designer is to integrate effectively goal-directed and reactive behavior.

Social ability:-
Intelligent agents are proficient of cooperating with other agents (and possibly humans), through negotiation and/or collaboration, to satisfy their design objectives [5], [6].

Proposed system:-
This technique was developed based on LFU policy with some significant improvements that let the web object near to end user to reduce the network traffic and increase the availability of web object in cache.
ALFUR technique used additional factors to select the victim- object to be deleted- and remove it from cache to free space for another object. All calculations to select the object to remove is done using multi-agent system.

Although ALFUR technique can be implemented in cache memory at end user side , proxy server side or original server side ; it has been implemented at proxy cache memory side; because proxy server caching is widely operated by computer network administrators to reduce user delays and to alleviate Internet overcrowding.

ALFUR technique common properties:-
1. Distributed Manner: The knowledge required to solve some problems does not reside in a single resource or agent, so that cooperation of many individual agents is needed to solve the problems.
2. Speed: Each agent has its own local processor and memory.
3. Efficiency: Not all knowledge is needed for all tasks, the agent uses the only part of the knowledge required to solve the problem.
4. Reliability: Multi-agent system is more reliable because there would be multiple agents in the setup which provide particular functionality or service. If an agent resource providing some functionality dies, another agent may take over.

ALFURTechnique:-
ALFUR properties rotate around three issues:-
First, it provides the characteristics of intelligent agent which have been discussed here before, beside this Agents in ALFUR Technique can act independently that states Autonomous. Each agent is independent with local tasks, it has the capabilities of problem-solving and decision-making and acts as a centralized management system.

Second, ALFUR considers four mechanisms in dealing with multi agents system:-
Cooperation: It is the process of sharing responsibilities in satisfying shared goals and generating dependent roles in joint activities.

Coordination: It is the process of management of agents' activities so that they coordinate their deeds with each other in order to share resources, meet their own interests.

Independence: every agent inALFUR technique is able to work concurrently and relatively independently, it has its own goal to increase hit ratio and byte hit ratio, but it is also capable of coordinating with other agents in order to achieve a common goal.

Agents Communication: Agents communicate among themselves by message passing, an agent can be permanently ready to receive messages from other agents and, at the same time, carry out its own computational tasks.

Third, ALFUR technique contains some artificial intelligence techniques such as:

Learning: learning provides an excellent method for optimizing agent's action. Reinforcement learning (RL) is a generic name given to a family of techniques in which an agent tries to learn a task by directly interacting with the environment. In multi-agent reinforcement learning, many agents are simultaneously learning by interacting with the environment and with each other.

There are two popular learning algorithms for single-agent systems: value iteration and Q-learning, and they can be extended to multi-agent systems.

ALFURDescription:-
ALFUR is a new multi agent technique that consists of four big agents: Reader agent, Analyzer agent, Removal agent and Performance agent, which are disused as following:

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

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

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

Performance evaluator Agent: calculates 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: 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 to take a decision.
2. The Removal Module: contains the parent cache removal agent that cleans up the cache according to web object frequencies, sizes, and times.

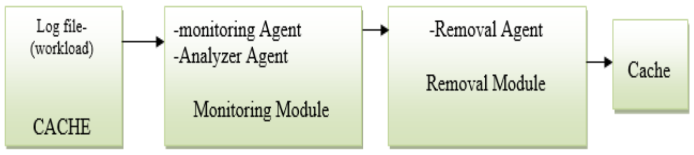

Figure1:- Model Architecture

AlgorithmMethodology:-
Proposed ALFUR new technique reads the objects in cache memory using a reader agent, then it analyzes the cache object using an analyzer agent that calculates average of frequencies, object's frequency, size and time, and then removes the object using a removal agent based on the following conditions:

If object's frequency is greater than the average of frequencies, then don't remove the object, otherwise remove the 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.

If object's frequency equals the average frequencies and the average equals to the object's size, then calculate the average of the web objects' time stamp, if it is less than the web object time stamp, don't remove the object, otherwise remove the object.

Finally, the performance agent calculates the number of hit ratio and number of byte hit ratio to measure the performance of this new technique.

ALFUR algorithms:-

- Read web object
- Calculate web object frequency, size ,request time
- Calculate Average of web object frequency, size and request time
- IF object Freq>Average objects Freaq THEN
- ((Don't remove object
- else
- Remove object)
- Else IF object Freq=Average objects and object size>average objects size THEN
- (Remove object
- else
- Don't remove object)
- Else IF object Freq=Average object and object size=average object size and object request time>average request time THEN
- (Don't remove object
- Else
- remove object))
- end

Performance Metrics:-

To accomplish the objective, cache replacement policy depends on several key metrics. Based on these performance metrics, we can compare the performance of different algorithms. These performance metrics play a very important role in web cache performance calculation. Such replacement policy aims to optimize performance metrics. To evaluate the performance of the cache, performance metrics are being used. The most commonly used are hit rate, byte hit rate saved bandwidth, and delay saving ratio.

Hit rate:-

The percentage of all object requests which are found in the cache instead of transferring from the requested server.

Byte hit:-

The percentage of all data that is transferred straight from the cache rather than from the requested server.

This paper measures the performance of proposed ALFUR model using Hit and Byte hit ratio for the new cache file after implement the model.

The hit ratio calculates the summation of requestedobjectsfound in new cache file divided by the number of all objects in original cache.

$$HitRatio = \frac{\sum_{i=1}^{n} \partial i}{n}$$

The byte hit ratio calculates the summation of requested objects size found in new cache file divided by total objects in original cache.

$$ByteHitRatio = \frac{\sum_{i=1}^{n} b_i \partial i}{\sum_{i=1}^{n} b_i}$$

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

Result:-

The following figures show the comparison of ALFUR with traditional LFU 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

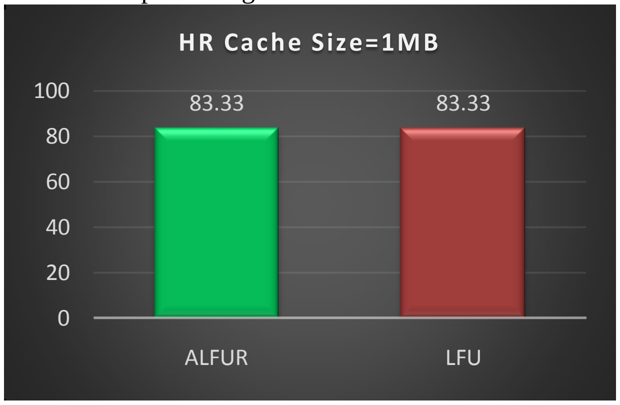

Figure 2:- HR (ALFUR vs. LFU cache size 1Mb)

Hit ratio Comparison figure 3 Tested with cache size 6MB

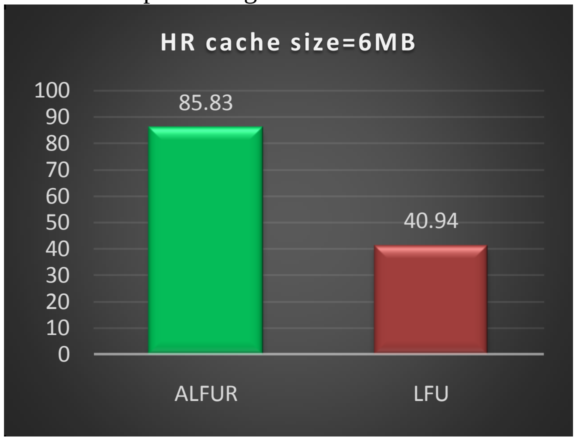

Figure 3:- HR (ALFUR vs. LFU cache size 6Mb)

Hit ratio Comparison figure 4 Tested with cache size 500MB

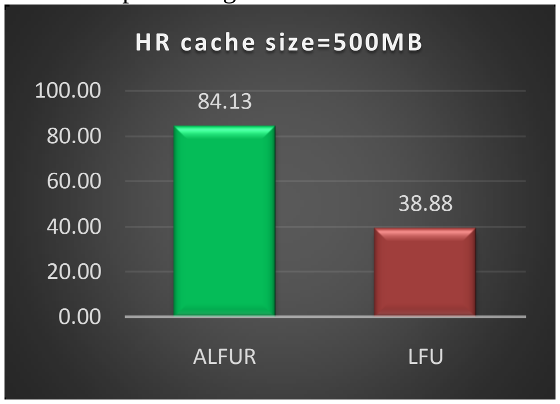

Figure 4:- HR (ALFUR vs. LFU cache size 500MB)

Hit ratio Comparison figure 5 Tested with cache size 800MB

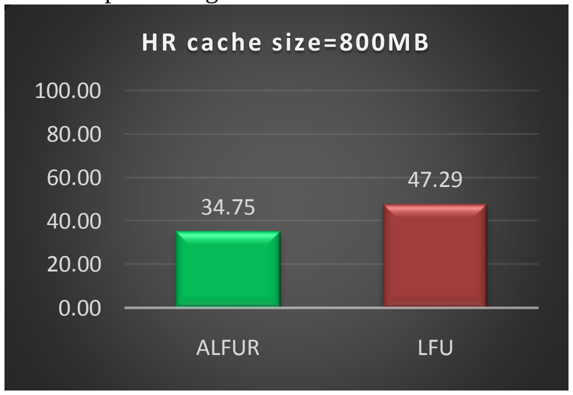

Figure 5:- HR (ALFUR vs. LFU cache size 800MB)

Hit ratio Comparison figure 6 Tested with cache size 1GB

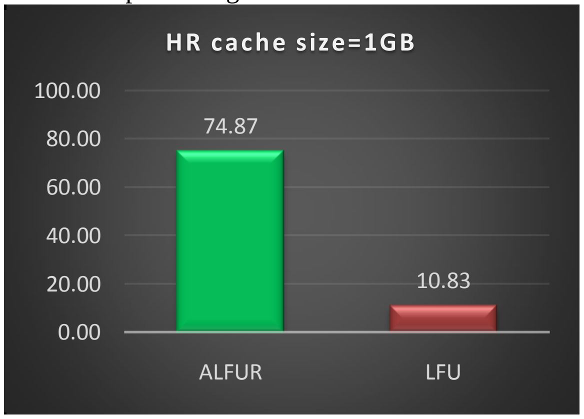

Figure 6:- HR (ALFUR vs. LFU cache size 1GB)

Byte Hit Ration Performance comparison (ALFUR vs. LFU):-

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

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

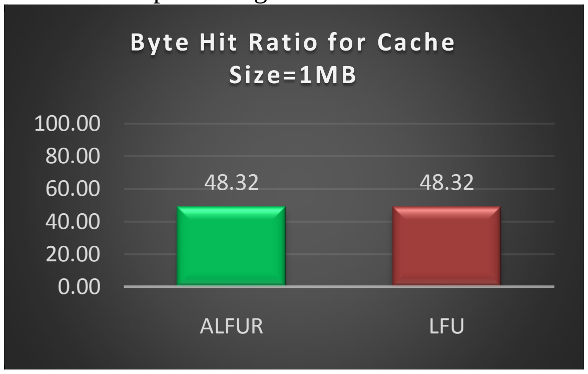

Figure 7:- BHR (ALFUR vs. LFU cache size 1MB)

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

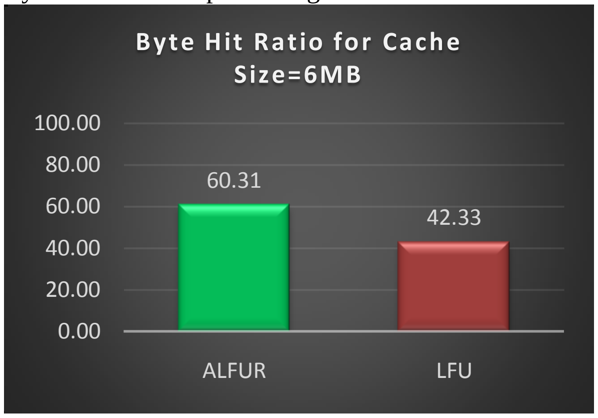

Figure 8:- BHR (ALFUR vs. LFU cache size 6MB)

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

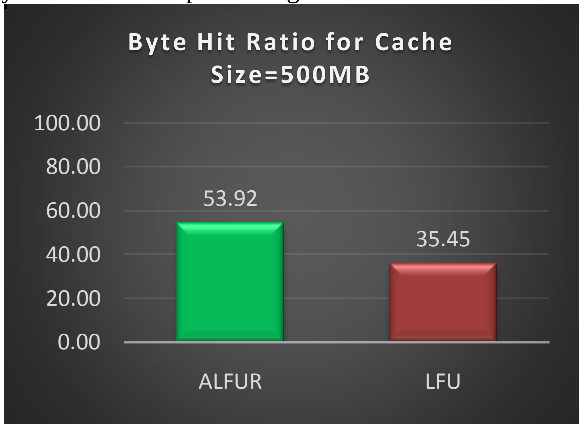

Figure 9:- BHR (ALFUR vs. LFU cache size 500MB)

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

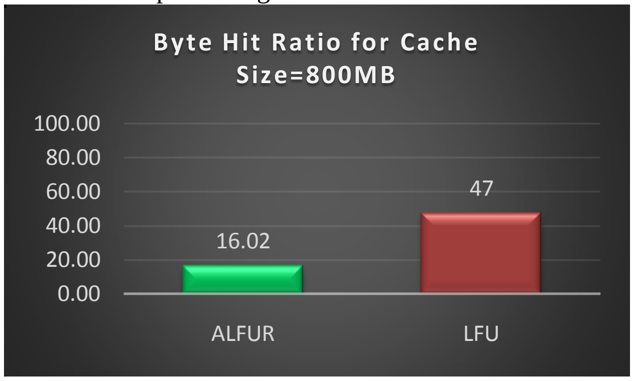

Figure 10:- BHR (ALFUR vs. LFU cache size 800MB)

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

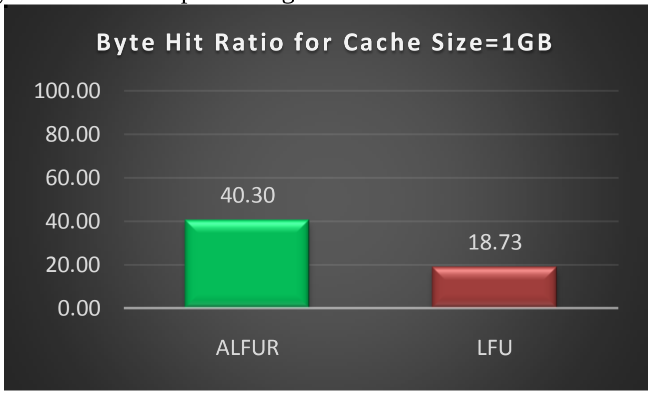

Figure 11:- BHR (ALFUR vs. LFU cache size 1GB)

ALFUR Best Result In Hit Ratio and Byte Hit Ratio:-

The figure 12 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.

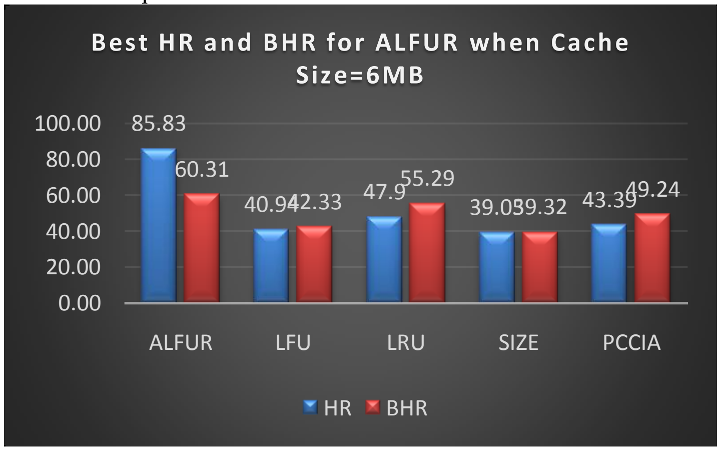

Figure 12:- Best HR and BHR for ALFUR

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

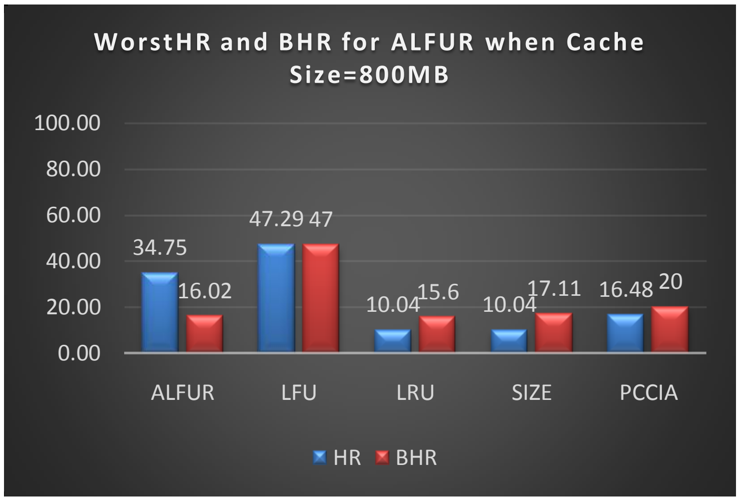

Figure 13:- Worst HR and BHR for ALFUR

Result Discussion:-

The webtraff simulation results illustrated that the cache size plays a main role in improving the performance of the web cache. There is a difference when the cache size increases and decreases .the new ALFUR algorithm has better performance over traditional LFU replacement algorithms in terms of Hit Rate and Byte Hit Rate.

Result Discussion in terms of Hit Ratio (ALFUR vs. LFU):-

This section consists of performance result discussion for data generated using webtraff in terms of Hit Ratio between ALFUR and LFU.

- Figure 2, cache size is equal to 1Mb the ALFUR and LFU are same result with hit ratio 83.33%. This rate of hit ratio is forth range for all samples.
- Figure 3, cache size is equal to 6Mb the ALFUR is in the better rate than LFU in this size with hit ratio 85.83% and LFU hit ratio is equal to 40.94%.ALFUR is better than LFU in this size with +44.89% rate of hit ratio. This rate of hit ratio is the third range for all samples.
- Figure 4, cache size is equal to 500Mb the ALFUR is in the better rate than LFU in this size with hit ratio 84.13% and LFU hit ratio is equal 38.88%.ALFUR is better than LFU in this size with +45.25% rate of hit ratio. This rate of hit ratio is the second range for all samples.
- Figure 5, cache size is equal to 800Mb the LFU is in the better rate than ALFUR in this size with hit ratio 47.29% and ALFUR hit ratio is equal to 34.745%.ALFUR is worse than LFU in this size with -12.54% rate of hit ratio. This rate of hit ratio is the fifth range for all samples.
- Figure 6, cache size is equal to 1 GB the ALFUR is in the better rate than LFU in this size with hit ratio 74.87% and LFU hit ratio is equal to 10.83%.ALFUR is better than LFU in this size with +64.04% rate of hit ratio. This rate of hit ratio is the first range for all samples.

Result Discussion in terms of Byte Hit Ratio (ALFUR vs. LFU):-

This section consists of performance result discussion for data generated using webtraff in term of Byte Hit Ratio between ALFUR and LFU.

- Figure 7, cache size is equal to 1Mb, the ALFUR and LFU are same result with byte hit ratio 48.32%. This rate of byte hit ratio is forth range for all samples.
- Figure 8, cache size is equal to 6Mb the ALFUR is in the better rate than LFU in this size with byte hit ratio 60.31% and LFU byte hit ratio is equal to 42.33%.ALFUR is better than LFU in this size with +17.98% rate of byte hit ratio. This rate of byte hit ratio is the third range for all samples.
- Figure 9, cache size is equal to 500Mb the ALFUR is in the better rate than LFU in this size with byte hit ratio 53.92% and LFU byte hit ratio is equal to 35.45%.ALFUR is better than LFU in this size with +18.47% rate of byte hit ratio. This rate of byte hit ratio is the second range for all samples.
- Figure 10, cache size is equal to 800Mb the LFU is in the better rate than ALFUR in this size with byte hit ratio 47.00% and ALFUR byte hit ratio is equal to 16.02%. ALFUR is worse than LFU in this size with -30.98% rate of byte hit ratio. This rate of byte hit ratio is the fifth range for all samples.

- Figure 11, cache size is equal to 1Gb the ALFUR is in the better rate than LFU in this size with byte hit ratio 40.30% and LFU byte hit ratio is equal to 18.73%.ALFUR is better than LFU in this size with +21.57% rate of byte hit ratio. This rate of byte hit ratio is the first range for all samples.

Conclusion:-

This paper introduces a new LFU web cache replacement using intelligent agent, this algorithm used more than one factors for removing object from cache memory. ALFUR compares between averages number of frequency with object request and removes the least frequency, if more than object have same frequency, it compares average object size with object size, and remove object with the biggest size, if more than object have same average object size, then finally it compares between time stamps of object and average of time stamp and remove the object with old request time. The comparison condition illustrated ALFUR algorithm. ALFUR is best performance when comparing Hit, Byte Hit, and Miss Ratio, The average ALFUR hit rate is equal to 72.58%, and from above result the ALFUR is better than average LFU which is equal to 44.25%. This means that ALFUR algorithm has better performance than LFU in terms of Hit Ratio with +28.33% rate. The average ALFUR byte hit rate is equal to 43.77%, from above result the ALFUR is better performance than average LFU which is equal to 38.37%. This mean that ALFUR algorithm has better performance than LFU in the terms of Byte Hit Ratio with +5.41% rate.

References:-

1. Waleed Ali, Siti Mariyam Shamsuddin, and Abdul Samad Ismail "A Survey of Web Caching and Prefetching",Universiti Teknologi Malaysia, ICSRS Publication, 2011.
2. Abdullah balamash and Marwan krunz, “An Overview of Web Caching Replacement Algorithms”, IEEE Communications Surveys &Tutorials, 2004, Vol.6 (2).
3. K.Geetha, Dr.N.Ammasai Gounden, Monikandan S.,”SEMALRU: An Implementation of modified web cache replacement algorithm”, IEEE,2009.
4. Kapil Arora, Dhawaleswar Rao ChWeb Cache Page Replacement by Using LRU and LFU Algorithms with Hit Ratio: A Case Unification, International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 3232 – 3235.
5. Rudowsky, Proceedings of the Americas Conference on Information Systems, New York, New York, August 2004
6. Rudowsky, Proceedings of the Americas Conference on Information Systems, New York, New York, August 2004
7. Hiba. A. Nasir, Yahia. A. Mohammed, Amir. A. Eisa, "Agent-based Proxy Cache Cleanup Model using Fuzzy Logic", proceedings of International Conference of Computing Electrical and Electronic Engineering (ICCEEE), Khartoum, Sudan, August 2013.