International Journal of Engineering Research and Advanced Technology (IJERAT)

DOI: 10.31695/IJERAT.2019.3583

E-ISSN : 2454-6135

Volume.5, Issue 12
December -2019

Subject Review: Comparing and Analyzing Page Fault Schemes According to Estimated Time

Salam A. Hussein[1] Mohsin R. Kareem[2] & Dena N. George[3]

[1-3] Assistant lecturers

[1-3]Department of Computer Science, Collage of Education, University of Mustansiriyah, Iraq

[2]Department of Computer Science, Collage of Basic Education, University of Mustansiriyah, Iraq

ABSTRACT

Active page replacement approaches are required for the virtual memory for the purpose of resolving what data pages are going to be driven out from the memory when a page fault occurs. For years, a lot of algorithms have been developed to avoid page fault, all these algorithms have the efforts of decreasing the proportion related to the reduce error in the same time as incurring minimal overhead. Since up-to-date patterns of memory access have been studied, the study has the aim of creating novel methods for page replacement that can be adaptive to the varying workloads. The presented review aims of comparing the assessed replacement time for the main methods of page replacement. Some conventional methods like CLOCK and LRU will be examined, as well as some new methods like 2-DPR, CLOCK-Pro, LIRS, and CAR.

Key Words: *Page fault, LRU, LIRS, CLOCK-Pro, ARC, CAR, 2-DPR.*

1. INTRODUCTION

To make use of multi-programming systems, it is of high importance to incorporate the implementation regarding additional programs which could be physically accommodated in the main memory. Therefore, 2-level memory hierarchy will be used in this study; this memory hierarchy contains faster, yet costly main memory and slower, yet low-cost secondary memory. The systems of the virtual memory are using this hierarchy for fetching program's fragments into the main memory from secondary memory with regard to certain units referred to as pages. The pages are going to be fetched to the main memory apart from the case when executing process ordered them; this can be referred to as demand paging. Page fault will happen in the case when referenced page isn't in the main memory and must be taken from the secondary memory. Existing page must be evicted in typical cases. Selecting pages is utilized via the methods of page replacement that are attempting to reduce page fault rate at least overhead [1].

2. RELATED WORK

(Belays' MIN) is considered as the optimum algorithm that ensures excellent performance. The optimum replacement algorithm indicated that in the case of full memory, page which is going to be un-referenced for the longest time will be evicted. Highest hit ratio resulted from this system. Obviously, it is impossible to achieve this policy, since the future events must be identified via the OS. Yet, it is serving as standard against which to assess applicable algorithms [2].

(LRU policy) is achieved according to the locality's principle which indicated that program and data references in processes have tendency towards cluster. LRU replacement policy select data page for replacement that wasn't referenced for longest time. For some time, LRU has been defined as optimal online approach. The challenge related to this method is that it is difficult to implement. One of the methods is tagging each one of data pages with time related to their last reference; this must be implemented at each one of the memory references, data and instruction. The policy of LRU is very efficient, yet there is a difficulty in implementing and imposing significant overhead [3].

(CLOCK algorithm) should be organized in circular list which represents page. The clock's hand is applied for pointing to buffer's oldest data page. Each one of the data pages has related reference bit which is fixed in the case when specific data page has been referenced. The policy of cache replacement policy will be invoked when there is a page fault, where the data page is going to be pointed to through hand (inspecting the oldest data page). In the case when setting the data page's reference bit, then, the bit is going to be reset to 0, also the next oldest data page will be pointed to via the hand. Such procedure will continue until

the data page that has reference bit 0 is detected. Therefore, data page will be eliminated from buffer; new data page will be inserted in its place while setting the reference bit to 0. It was indicated that CLOCK algorithm is similar to LRU as it provides high efficiency and least overhead [4].

(Dueling CLOCK) can be defined as adaptive replacement policy which functions according to CLOCK algorithm. One of the main drawbacks related to conventional CLOCK algorithm is that it cannot be considered as scan-resistant (The algorithms of cache replacement which doesn't enable scanning to move the regularly utilized data page out of the main memory is indicated as scan resistant [2]). For overcoming such disadvantage, scan-resistant CLOCK algorithm will be developed, also set dueling method will be applied for adaptively selecting scan-resistant CLOCK or CLOCK as main algorithm used for cache replacement. For the purpose of achieving CLOCK scan-resistant, the single required modification is that the clock's hand must be pointing to buffer's newest data page instead of oldest one. It can be considered as an approach of adaptive replacement that uses the structure of CLOCK. Cache is going to be divided into 3 sets, (M1), (M2) and (M3). The CLOCK algorithm will always be implemented as replacement method for the small set (M1), while the scan-resistant CLOCK will always be used by the small set (M2). Large set (M3) could be applied between scan-resistant CLOCK and CLOCK based on comparative performance related to the former 2 sets. For the purpose of specifying the replacement policy that should be applied for set (M3), (ten-bit) policy select counter (PSEL) will be implemented [5].

Low Inter-reference Recency Set algorithm (LIRS) considers data pages' Inter-Reference Recency as significant aspect for eviction. The data pages' Inter-Reference Recency (IRR) indicates number of the other data pages which are accessed between 2 successive references to the data page. There is a suggestion that in the case when the data page's IRR is large, thus, the next IRR related to page will be possible also large and therefore data page will be proper for eviction in accordance with Belady's MIN. It should be indicated that the high IRR-data pages which are chosen for eviction are applied lately. Data pages with low-IRR (LIR) and data pages with high-IRR (HIR) are distinguished via the algorithm. The number of the HIR and LIR data pages will be selected in a way that small percentage of HIR and all LIR data pages are kept in cache. When page fault exists, resident HIR data pages that have the highest recency will be eliminated from cache, requested data pages are going to be brought in, in the case when new IRR related to requested data pages is considered to be small in comparison to recency related to certain LIR data page, at that time their LIR/HIR statuses will be interchanged. Typically, approximately 1 percent related to cache will be applied for storing HIR-data pages, whereas 99 percent related to cache will be kept for the LIR-data pages [6].

(CLOCK-Pro) algorithm, as can be seen in figure-1, attempt to use CLOCK for approximating LIRS. Reuse distance, that is considered to be analogous to the IRR of LIRS, is a parameter of high importance to the page replacement decision in the CLOCK-Pro. In the case of accessing a page, the page will be considered as "hot" page in the case when it has little distance of reuse, while it will be considered as "cold" page in the case when it has huge distance of reuse. Pages will be accessed in a list where the pages with small recency will be placed at list's head, while pages with large larger recency will be placed at the list's end. Circular list will be used to keep the cache page entries. There are 3 status bits related to such pages; hot or cold indicator, referencing bit, as well as indicator for each cold page which might specify if a page is in test period. There are 3 hands in CLOCK-Pro approach. ($HAND_{cold}$) is considered to be the first one which is pointing to last resident cold page, or cold page which will be replaced next. Throughout the page replacement process, page that is selected through ($HAND_{cold}$) will be evicted in the case when reference bit related to the page is 0. ($HAND_{hot}$) is the second hand which point to the hot page with greatest recency. In the case when the reference bit related to the page that is selected via ($HAND_{hot}$) is 0, the page will be indicated as cold page. In the case when reference bit is 1, at that point it will become 0 and the hand will be moving clockwise by one page. Finally, ($HAND_{hot}$) will stop at hot page, while the ($HAND_{test}$) will stand at next cold page. In the case when there is a page fault, the page will be assessed in memory. When the missed page is in list as a cold page, it will be transformed to hot page; also it will be moved to top of list of hot pages. Furthermore, ($HAND_{hot}$) will be run to changing hot page which has greatest recency to the cold page. In the case when missed page isn't in list, it will be fetched in memory and set as cold page. The page will be placed at top of the list related to the cold pages and its Probationary period will be initiated as can be seen in the figure-1[7-8].

DOI : 10.31695/IJERAT.2019.3583

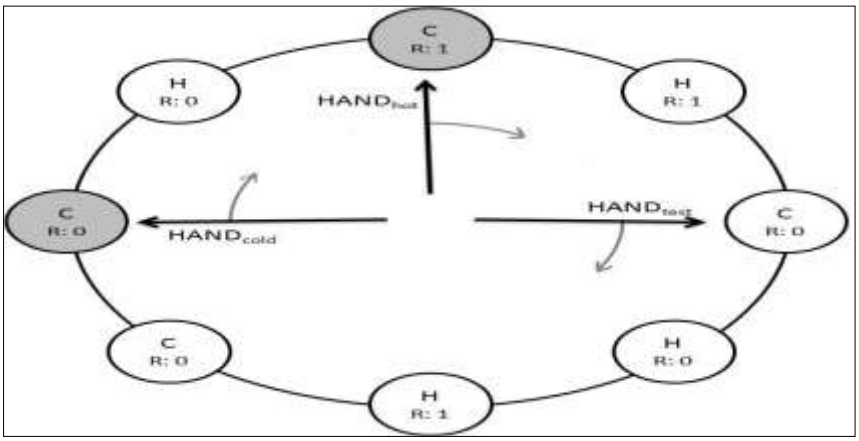

Figure-1 Working of CLOCK-Pro

(CAR algorithm) have the aim of merging ARC's adaptive procedure with CLOCK's implementation effectiveness. There are 4 doubly linked lists (T1, T2, B1, and B2) are maintained in the algorithm. B1 as well as B1 are considered as simple LRU lists, while T1 and T2 are CLOCKs. The model related to the lists is comparable to that of the ARC. Furthermore, the lists T1 in addition to T2 i.e. data pages in cache have reference bit which could be reset or set. Naturally T10, B1 specify "recency" as can be seen in the (Figure-2), while T11 U *T*2 U *B*2 specifies "frequency" as can be seen in the (Figure-3). The accurate definition related to the 4 lists is:

1. *T*10 as well as *B*1 consists of all data pages which have been exactly referenced one time from its latest eviction from *T*1 U *T*2 U *B*1 U *B*2 or on no occasion referenced previously from its inception.
2. *T2*, *B2*, and *T11* consists of all data pages which have been referenced over one time starting from its latest eviction from *T*1 U *T*2 U *B*1 U *B*2. The 2 significant constraints related to sizes of T1, T2, B1 and B2 are [10].

1. $0 \leq |T1|+|B1| \leq c$. Generally, *T*1 U *B*1 captures recency. Size related to regularly accessed and recently accessed data pages continue to change. This will prevent the data page that is accessed just one time from taking up the whole cache directory that has a size of (2*c*) due to the fact that the increase in the size related to *T*1 U *B*1 specify that recent referenced data pages aren't referenced another time and that indicates that the stored recency data is not useful. Therefore, it is indicating that just the regularly utilized data pages will re-referenced or new data pages will be referenced.
2. $0 \leq |T2|+|B2| \leq 2c$. In the case when just set of data pages will be accessed regularly, no new references exist. Cache directory hold information related merely to the frequency [9] [10].

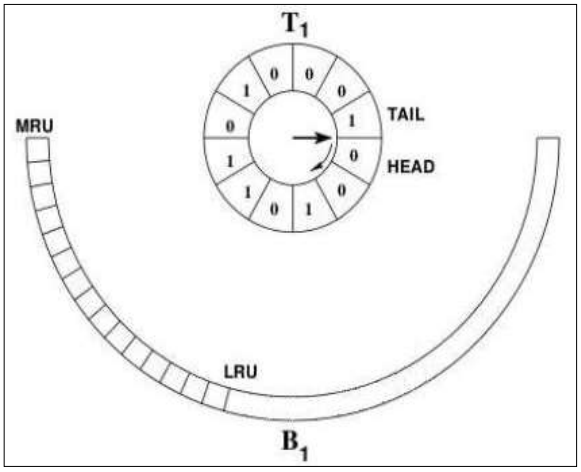

Figure-2 Recency

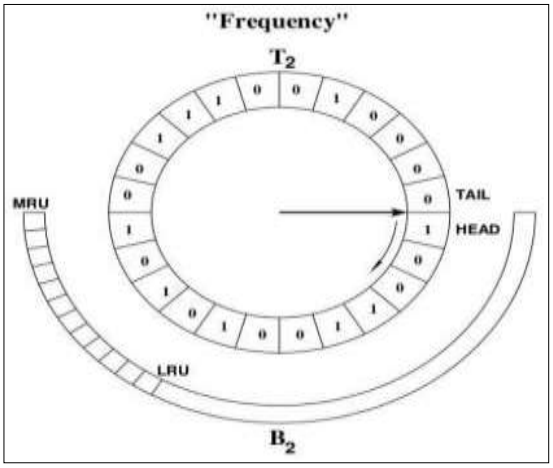

Figure-3 Frequency

The fundamental idea of this algorithm may be explained as follows, In the instance of a cache hit, the page of data which has been requested is fetched from cache and the reference bit which is associated to it is specified. Rather than this, in case where the page doesn't reside in cache, and instead it resides in the B-1 list, this will indicate the fact that the data page was utilized once lately prior to the eviction from cache. This page is transferred afterwards to the head of the T-2 and the reference bit of that page is given the value of zero. In addition to that, a hit in the B1 points to the fact that the data pages that have been utilized once lately are needed again, which implies a workload of recency favoring. Which is why, p value must increase, which results in increasing T-1 size. Which is why, in the case where the page doesn't reside in the cache and instead in the B-2, it will

DOI : 10.31695/IJERAT.2019.3583

indicate the fact that the page was utilized often, prior to the eviction from cache? This page is transferred afterwards to the head of the T-2 and the reference bit of that page is given the value of zero. Moreover, a hit in the B-2 will indicate the fact that pages which are utilized often are requested again, which implies a workload of frequency favoring. This is why p's value must decrease, which results in increasing T-2 size. Lastly, in the case where the data page hasn't been found in the B-1 U B-2, then the page is added to the position of the MRU in the T-1. In any case from above, in the case where cache is full (i.e. |T1| + |T2| = c), then the CLOCK policy is implemented on either T-1 or T-2 based on p parameter [11].

(2-DPR algorithm) is the most modern replacement algorithm around the world. It was proposed in (2016) by Safana H. Abbas and Salam A. Hussein. 2DPR, as in fig.4, includes 4 levels, which are (L-1, L-2, L-3 and L-4), representing actual size of the cache (fig.3). A page inserting is performed via base level (i.e.L-1). Where an eviction of a page is carried out via the top level (i.e. L-4) that has to contain a single page size. Each page in the system is related to 2 counters, which are:

1. **A reference counter (R):** This is responsible for capturing "recency" of each page residing in the cache and operates in the following manner:

- R is equal to R+1 for pages which reside in the cache and that haven't been referenced.
- R is equal to Zero for pages that have been referenced or reside in the cache and re-referenced.

2. **Frequency counter (F):** reflects "frequency" value for every page residing in the cache and it operates in the following manner:

- F is equal to F for all pages residing in the cache and that haven't been referenced.
- F is equal to F+1 for the pages that have been referenced or the ones that reside in the cache and have been re-referenced.

Frequency counter is under the control of a threshold, whose value is specified by the designers of the system. Initially, the following steps will be performed:

1. in the case of the occurrence of a page error, and there has been an empty space in (L-1). New referenced page is going to be obtained from the M.M and inserted in the (L-1) and existing and referenced page counters are going to be organized in the following manner:

- Existing pages: R=R+1 / F is equal to F.
- Referenced page: R is equal to Zero / F=F+1 (holding a preliminary value = 0).

2. In the case of the occurrence of a page error, and there weren't any empty spaces in the (L-1) but the (L-2) includes a space which is empty, existing page counters are going to be organized as: R=R+1/ F=F.

In this case, the [MAXR] page in the (L-1) is going to be moved from the (L-1) to the (L-2). The new referenced page is going to be obtained from the M.M and inserted into the (L-1) and the referenced page counters are going to be organized in the following manner:

R=0 / F=F+1 (holding a preliminary value = 0).

3. In the case of the occurrence of a page error, and there weren't any empty spaces in neither one of (L-1) nor (L-2) but (L-3) had an empty space, the existing page counters are going to be organized in the following manner: R=R+1/ F=F.

In this case, the [MAXR] page in the (L-2) is going to be moved from the (L-2) to the (L-3), and the [MAXR] page in the (L-1) is going to be moved from the (L-1) to the (L-2). The new referenced page is going to be obtained from the M.M and inserted into the (L-1) and the referenced page counters are going to be organized in the following way:

R=0 / F=F+1 (holding a preliminary value which is equal to 0).

4. If a page fault occurrence and there weren't any empty spaces in neither one of (L-1), (L-2) nor (L-3) but (L-4) had an empty space, the existing page counters are going to be organized in the following manner: R=R+1 / F=F.

In this case, the [MAXR] page in the (L-3) is going to be moved from the (L-3) to the (L-4), the [MAXR] page in the (L-2) is going to be moved from the (L-2) into the (L-3) and the [MAXR] page in the (L-1) is going to be moved from the (L-1) to the (L-2). The new referenced page is going to be taken from the M.M and inserted into the (L-1) and the referenced page counters are going to be organized in the following manner:

R=0 / F=F+1 (holding a preliminary value which is equal to 0).

5. and lastly, in the case of the occurrence of a page fault and there weren't any empty spaces in neither (L-1), (L-2), (L-3) nor (L-4), meaning that the cache is full and a single page has to be evicted for the sake of releasing a space into the new referenced page, after that, the (L-4) page is going to be evicted (due to the fact that it is going to be the optimal option for evicting, due to the fact

DOI : 10.31695/IJERAT.2019.3583

that it was least frequently as well as least recently used). The existing pages' counters are going to be organized in the following manner: R=R+1 / F=F.

After that, the [MAXR] page in the (L-3) is going to be moved from the (L-3) into the (L-4), the [MAXR] page in the (L-2) is going to be moved from the (L-2) to the (L-3) and the [MAXR] page in the (L-1) is going to be moved from the (L-1) to the (L-2). The new referenced page is going to be obtained from the M.M and inserted into the (L-1) and the referenced page counters are going to be organized in the following manner:

R=0 / F=F+1 (holding a preliminary value which is equal to 0).

Cache hit indicates the fact of when CPU makes a request for an item of data residing in the cache.

1. in the case where the data page exists, and the page which has been re-referenced exist in (L-1), in this case the only action which is going to will be taken is updating re-referenced and existing page counters in the following manner:

- Existing pages: R=R+1 / F=F.
- Re-referenced page: R=0 / F=F+1.

2. in the case where the data page exists, and the re-referenced page exists in the (L-2), then the existing and re-referenced page counters are going to be organized in the following manner:

- Existing pages: R=R+1 / F=F.
- Re-referenced page: R=0 / F=F+1.

After that, the [MAXR] page in the (L-1) id going to be exchanged with the re-referenced page.

3. in the case where the data page exists, and the re-referenced page exists in the (L-3), in this case, the re-referenced and existing page counters are going to be organized in the following manner:

- Existing pages: R=R+1 / F=F.
- Re-referenced page: R=0 / F=F+1.

After that a choice of three is going to be made based on the conditions below:

A. in the case where the re-referenced page's frequency counter (i.e. the Fcounter) was ($1 \leq F \leq 3$), then the [MAXR] page in the (L-2) is going to be exchanged with the re-referenced page.

B. in the case where the re-referenced page's frequency counter was ($4 \leq F \leq 6$), then the [MAXR] page in the (L-1) is going to be moved from the (L-1) to the (L-2) and the [MAXR] page in the (L-2) is going to be moved from the (L-2) to the (L-3). Then the re-referenced page is going to be placed in the (L-1).

C. in the case where the re-referenced page's frequency counter was ($F>6$), then the [MAXR] page in the (L-2) is going to be exchanged with the re-referenced page. And the counters of the re-referenced page are going to be organized in the following manner: [R=0/F=1].

4. in the case where the data page is existent, and re-referenced page is residing in the (L-4), then the re-referenced and existing page counters are going to be organized in the following manner:

- Existing pages: R=R+1 / F=F.
- Re-referenced page: R=0 / F=F+1.

After that, 1 choice of 4 is going to be made based on the conditions below:

A. in the case where the re-referenced page's Frequency counter was ($1 \leq F \leq 2$), then the [MAXR] page in the (L-3) is going to be exchanged with the re-referenced page.

B. in the case where the re-referenced page's Frequency counter was ($3 \leq F \leq 4$), the [MAXR] page in the (L-2) is going to be moved from the (L-2) to the (L-3) and the [MAXR] page in the (L-3) is going to be moved from the (L-3) to the (L-4). After that, the re-referenced page is going to be located in the (L-2).

C. in the case where the re-referenced page's Frequency counter was ($5 \leq F \leq 6$), the [MAXR] page in the (L-1) is going to be moved from the (L-1) to the (L-2) and the [MAXR] page in the (L-2) is going to be moved from the (L-2) to the (L-3) and the [MAXR] page in the (L-3) is going to be moved from the (L-3) to the (L-4). After that, the re-referenced page is going to be moved to the (L-1).

D. in the case where the re-referenced page's Frequency counter was ($F>6$), then the [MAXR] page in the (L-3) is going to be exchanged with the re-referenced page. The counters of the re-referenced page are going be organized in the following manner: [R=0/F=1].

DOI : 10.31695/IJERAT.2019.3583

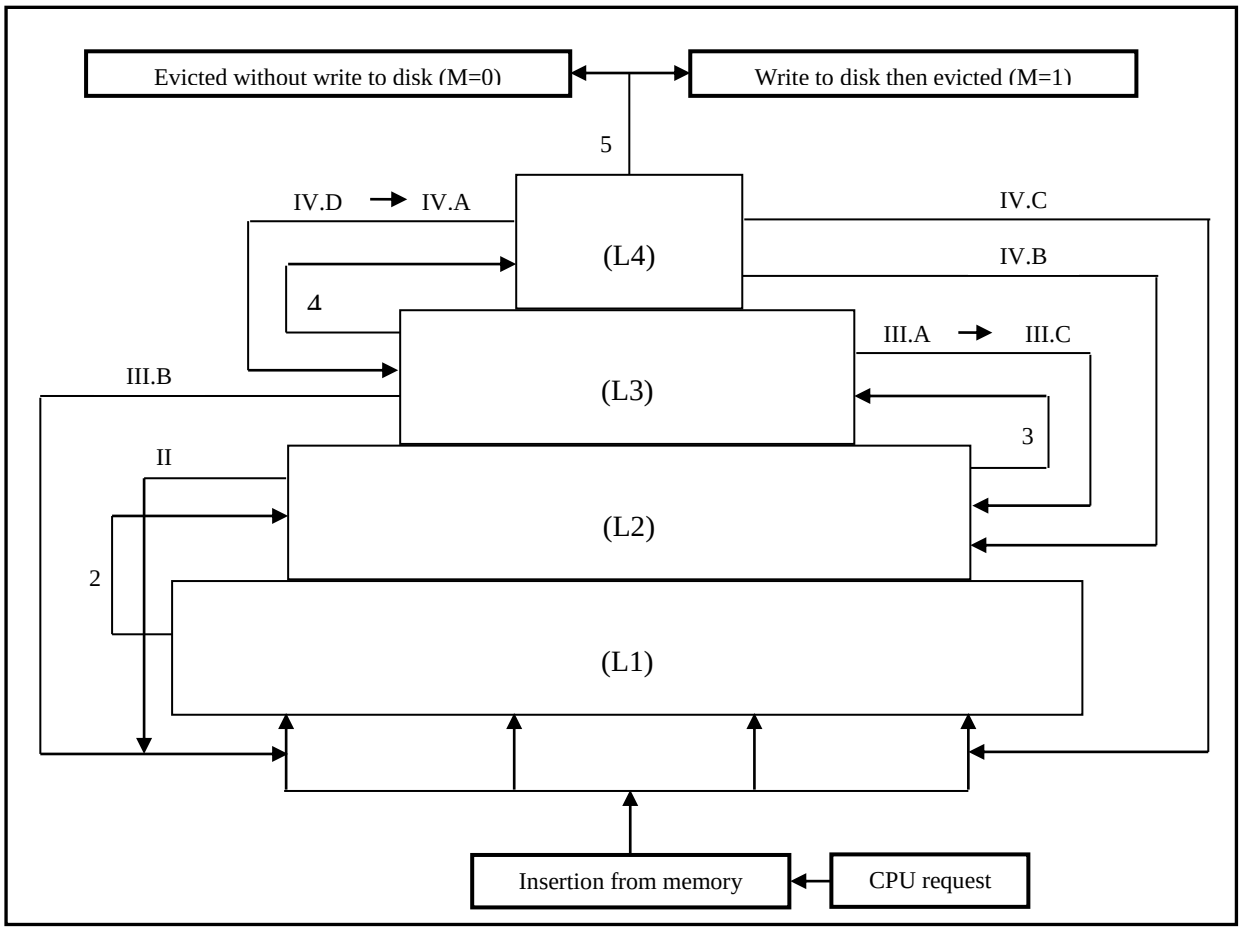

Figure-4 the 2-DPR algorithm

3. PERFORMANCE ANALYSIS

One of the most efficient criterions that have been utilized for judging the efficiency of the above mentioned techniques is estimated time measured in nano-seconds (N.S).

Thus, some of the previous techniques came back with the following results (as in table-1 and figure-5):

Table-1 table of estimated time measured in (N.S)

Cache size	LRU	LFU	LIRS	CAR	Clock-pro	2-DPR
7	36.65	26.20	31.78	43.51	36.05	43.03
8	38.89	33.00	32.00	39.88	33.74	45.23
11	43.96	35.01	39.02	45.90	40.57	46.06
14	47.00	39.78	44.16	44.98	42.03	47.15
23	45.31	41.57	47.37	59.97	42.51	53.73

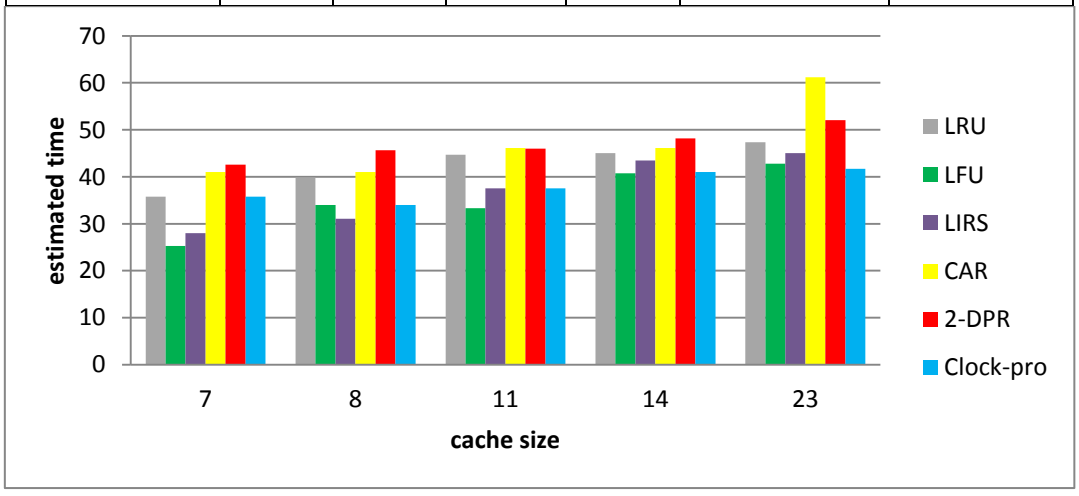

Figure-5 flowchart of hit ratio percentages

DOI : 10.31695/IJERAT.2019.3583

IV. CONCLUSIONS

According to the previous results, modern algorithms (algorithms with more than one factor) are slower than the classical algorithms (algorithms with only one factor) because multiple factors needs more time to decide which page is the best candidate for eviction.

REFERENCES

[1] Hassidim, A., "Cache Replacement Policies for Multicore Processors," ICS, 2010, pp. 501–509.

[2] Marty, M. R., "Cache Coherence Techniques for Multicore Processors," Ph.D, UNIVERSITY OF WISCONSIN - MADISON, 2008.

[3] Katz, R. H., Eggers, S. J., Wood, D. A., Perkins, C. L., and Sheldon, R. G., "Implementing a Cache Consistency Protocol," in Proceedings of the 12th Annual International Symposium on Computer Architecture, Los Alamitos, 1985, pp. 276–283.

[4] Denning, P. J., and Ullman, J. D., "Principles of Optimal Page Replacement," Journal of the Association for Computing Machinery, 1971, Vol. 18, pp. 80-93.

[5] Shamsheer Daula, S.M., Sreenivasa Murthy, K.E., and Amjad Khan, G., "A Throughput Analysis on Page Replacement Algorithms in Cache Memory Management," International Journal of Engineering Research and Applications (IJERA), 2012, Vol. 2, pp. 126-130.

[6] Swain, D., Dash, B. N., O. Shamkuwar, D., and Swain, D., "Analysis and Predictability of Page Replacement Techniques towards Optimized Performance," International Conference on Recent Trends in Information Technology and Computer Science (IRCTITCS), 2011, pp. 12-16.

[7] Al-Zoubi, H., Milenkovic, A., and Milenkovic, M., "Performance Evaluation of Cache Replacement Policies for the SPEC CPU2000 Benchmark Suite," Proc. 42nd ACM Southeast Conference, 2004, pp. 267-272.

[8] Arpaci-Dusseau, R. H., and Arpaci-Dusseau, A. C., Operating Systems: Three Easy Pieces: version 0.80, Wisconsin: Arpaci-Dusseau Books, Inc, 2014, pp. 227-244.

[9] Chavan, A. S., Nayak, K. R., Vora, K. D., Purohit, M. D., and Chawan, P. M., "A Comparison of Page Replacement Algorithms," IACSIT International Journal of Engineering and Technology, 2011, Vol. 3, NO.2, pp. 171-174.

[10] Godavarthy, A., Lakshminarasimhachar, S., and Gopinathan, S., "Simulation and Analysis of Cache Replacement Algorithms," Proc. International Conference on Computer Design, 2010, pp. 1-41.

[11] Hussein S. A. and Kareem M. R., " Compression and Analysis between Classic and Modern Cache Replacement Techniques" Iraqi Journal of Information Technology (IJIT), **2018**, Vol. 9, No:1, pp. 28-47.

[12] Abbas S. H. and Hussein S. A., "2-DPR: A Novel, High Performance Cache Replacement Algorithm," Diyala Journal For Pure Sciences (DJPS), July **2016**, Vol. 12, No:3, pp. 98-113.

[13] Hussein S. A., " A Proposed Multilevel Replacement Algorithm for Cache Memory," M.Sc, Al-Mustansiriyah University, **2016**.

DOI : 10.31695/IJERAT.2019.3583