International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-10, August 2020

Critical Scrutiny of Page Replacement Algorithms: FIFO, Optimal and LRU

Jogamohan Medak, Partha Pratim Gogoi

Abstract: ***Virtual memory plays an important role in memory management of an operating system. A process or a set of processes may have a requirement of memory space that may exceed the capacity of main memory. This situation is addressed by virtual memory where a certain memory space in secondary memory is treated as primary memory, i.e., main memory is virtually extended to secondary memory. When a process requires a page, it first scans in primary memory. If it is found then, process continues to execute, otherwise a situation arises, called page fault, which is addressed by page replacement algorithms. This algorithms swaps out a page from main memory to secondary memory and replaced it with another page from secondary memory in addition to the fact that it should have minimum page faults so that considerable amount of I/O operations, required for swapping in/out of pages, can be reduced. Several algorithms for page replacement have been formulated to increase the efficiency of page replacement technique. In this paper, mainly three page replacement algorithms: FIFO, Optimal and LRU are discussed, their behavioural pattern is analysed with systematic approach and a comparative analysis of these algorithms is recorded with proper diagram.***

Keywords: Belady's anomaly, FIFO, hit ratio, LRU, Optimal, page fault, Virtual memory.

I. INTRODUCTION

In a multiprogramming environment, several processes are executed simultaneously, i.e., processes, which are to be executed, must be available in primary memory. However, primary memory in a computer system is limited. If the executing processes require more memory space than available in the primary memory, then this situation is handled by a concept called virtual memory. Virtual memory is a memory management procedure, where a memory space, allocated in the secondary memory, acts as primary memory on a temporary basis. Virtual memory (logical address space) is divided into a fixed-length adjacent block of memory, called a page and its equivalent memory space in primary memory (physical address space) is called a frame. When a process requires a page for execution and it is not found in primary memory, then a page fault occurs. The required page has to be brought into the primary memory. However, if the primary memory has not enough space to accumulate the required page, then an existing page in primary memory should have to be replaced with the required page. A page replacement strategy decides which page needs to be swap out to replace the required page.

Revised Manuscript Received on August 30, 2020.
* Correspondence Author
Jogamohan Medak*, MCA, Tezpur University, Assam, India. E-mail: joga60mca@gamil.com
Partha Pratim Gogoi, MCA, Tezpur University, Assam, India. E-mail: ppgadmn@gmail.com

Minimum number of page fault results in a higher performance in execution, as a page fault has to be addressed by performing disk I/O operations that costs a lot of time.

II. PAGE REPLACEMENT ALGORITHMS

This paper will primarily concentrate on some commonly used page replacement algorithms and analyse them with number of page faults with different frame size. It will also take account of page hit ratio, i.e., the ratio of number of desired pages found (hit) in primary memory and the total number of pages demanded by a process. A higher hit ratio denotes greater efficiency of the page replacement algorithm.

A. First In, First Out (FIFO) Algorithm

This algorithm is the simplest technique of all the page replacement algorithms. It can be implemented by using a queue. The recently arrived page is positioned at rear, while the existing oldest page is positioned at front. When a page fault occurs, the page at front is chosen to be swapped out and recent page is swapped in and placed at rear. An algorithm for FIFO page replacement procedure is given below:

1. Start traversing the pages until all pages are visited.
   A. If frame is empty // *FIFO operation is performed using queue*
      a. Insert the pages into frame until the frame is full. //initially frame is empty.
      b. Increment the page fault in every insertion of page.
   B. Else
      a. If demanded page is present in the frame // no page fault
         i. Increment hit count.
      b. Else // page fault occur
         i. Replace the page in the front of the frame with the new page in the string.
         ii. Increment the page fault.
2. Return page fault & hit count.

Now, to elaborate this algorithm, let assume a reference string, 0 1 2 3 0 1 4 0 1 2 3 4 using frame size of 3, 4 and 5.

References	0	1	2	3	0	1	4	0	1	2	3	4
Frame	0	0	0	3	3	3	4	4	4	4	4	4
		1	1	1	0	0	0	0	0	2	2	2
			2	2	2	1	1	1	1	1	3	3
Observation	Page fault	Page fault	Page fault	Page fault	Page fault	Page fault	Page fault	Hit	Hit	Page fault	Page fault	Hit

Fig.1 FIFO page replacement algorithm using frame size 3

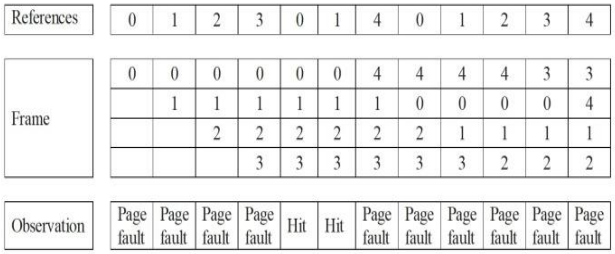

Fig.2 FIFO page replacement algorithm using frame size 4

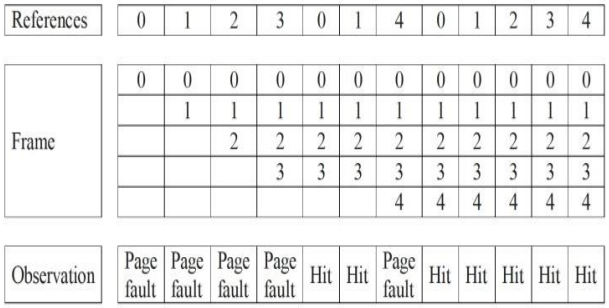

Fig.3 FIFO page replacement algorithm using frame size 5

A table is formed based on observation made above.

	Frame Size 3	Frame Size 4	Frame Size 5
Number of Hits	3	2	7
Number of Page Faults	9	10	5
Hit Ratio	0.25	0.17	0.58

Table.1 Analysis of FIFO page replacement algorithm

Generally, increase in the number of frame size reduces the number of page faults. However, in this algorithm, it is observed that page fault increases when frame size is increased from 3 to 4. This inconsistency is known as Belady's anomaly, where in certain condition, page fault increases with the increase in frame size. It swaps out a page that may be immediate next requirement due to lack of prediction ability based on past or future references.

B. Optimal Page Replacement Algorithm

This algorithm is considered to be the best page replacement algorithm, as it generates the lowest page fault rate of all the algorithms. This algorithm replaces the page that will not be used in near future or will only be used in distant future. An algorithm for optimal page replacement procedure is given below:

1. Start traversing the pages until all pages are visited.

A. If frame is empty.

a. Insert the pages into the frame until frame is full. //initially frame is empty.

b. Increment the page fault in every insertion of page.

B. Else

a. If demanded page is present in the frame. // no page fault

i. Increment hit count.

b. Else // page fault occur

i. Find the page in the frame that is farthest in future or never be used in future.

ii. Replace that page with new page.

iii. Increment page fault.

2. Return page fault & hit count.

This algorithm is exhibited below using the same reference string, 0 1 2 3 0 1 4 0 1 2 3 4 for frame size of 3, 4 and 5.

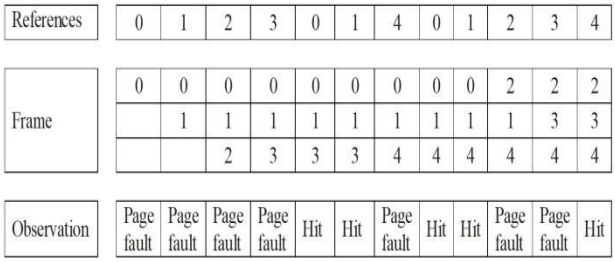

Fig.4 Optimal page replacement algorithm using frame size 3

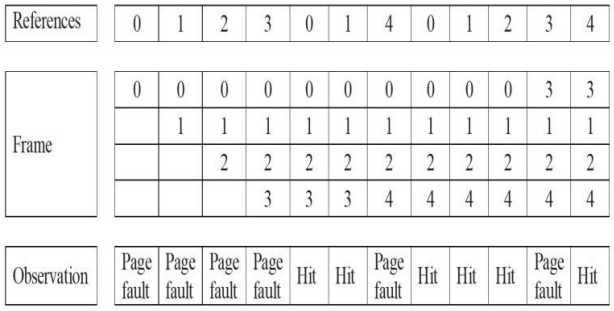

Fig.5 Optimal page replacement algorithm using frame size 4

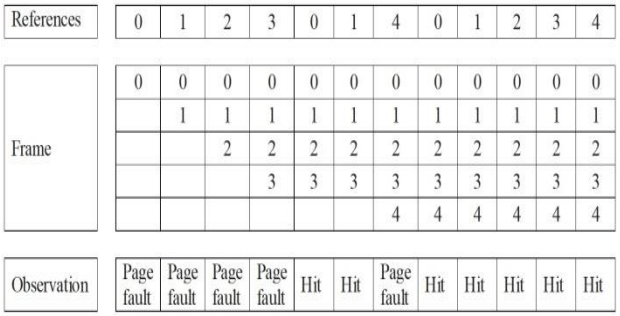

Fig.6 Optimal page replacement algorithm using frame size 5

Summary from the above observation for optimal page replacement algorithm is tabled below.

	Frame Size 3	Frame Size 4	Frame Size 5
Number of Hits	5	6	7
Number of Page Faults	7	6	5
Hit Ratio	0.42	0.5	0.58

Table.2 Analysis of Optimal page replacement algorithm

From the above table, it is observed that number of page fault decreases with the increase in frame size. However, this algorithm needs a reference to the future occurrence of the pages, which makes it very difficult to implement because operating system does not make exact prediction which pages will be used in near future.

C. Least Recently Used (LRU) Algorithm

Least Recently Used algorithm is one of the efficient algorithms. It replaces the page which has not been used for a long time. It is based on the assumption that pages that are recently used may be in use again in near future. It tries to predict the future occurrences based on the past activities. An algorithm for LRU page replacement procedure is given below:

1. Start traversing the pages until all pages are visited.

A. If frame is empty

a. Insert the pages into the frame until frame is full. //initially frame is empty.

b. Simultaneously maintain the recently inserted page in an index array equal to frame size. // Index array is used to keep track recently used page.

c. Increment the page fault in every insertion of page.

B. Else

a. If demanded page is present in the frame // no page fault

i. Increment hit count.

b. Else // page fault occur

i. Find the page in the frame that was least recently used by using index array.

ii. Replace that page with the demanded page.

iii. Update index with new page.

iv. Increment the page fault.

2. Return page fault & hit count.

Figures below help to understand this algorithm using the same reference string, 0 1 2 3 0 1 4 0 1 2 3 4 with frame size of 3, 4 and 5.

References	0	1	2	3	0	1	4	0	1	2	3	4
Frame	0	0	0	3	3	3	4	4	4	2	2	2
		1	1	1	0	0	0	0	0	0	3	3
			2	2	2	1	1	1	1	1	1	4
Observation	Page fault	Page fault	Page fault	Page fault	Page fault	Page fault	Page fault	Hit	Hit	Page fault	Page fault	Page fault

Fig.7 LRU page replacement algorithm using frame size 3

References	0	1	2	3	0	1	4	0	1	2	3	4
Frame	0	0	0	0	0	0	0	0	0	0	0	4
		1	1	1	1	1	1	1	1	1	1	1
			2	2	2	2	4	4	4	4	3	3
				3	3	3	3	3	3	2	2	2
Observation	Page fault	Page fault	Page fault	Page fault	Hit	Hit	Page fault	Hit	Hit	Page fault	Page fault	Page fault

Fig.8 LRU page replacement algorithm using frame size 4

References	0	1	2	3	0	1	4	0	1	2	3	4
Frame	0	0	0	0	0	0	0	0	0	0	0	0
		1	1	1	1	1	1	1	1	1	1	1
			2	2	2	2	2	2	2	2	2	2
				3	3	3	3	3	3	3	3	3
							4	4	4	4	4	4
Observation	Page fault	Page fault	Page fault	Page fault	Hit	Hit	Page fault	Hit	Hit	Hit	Hit	Hit

Fig.9 LRU page replacement algorithm using frame size 5

From the above observation, a table is constructed to analyse the data.

	Frame Size 3	Frame Size 4	Frame Size 5
Number of Hits	2	4	7
Number of Page Faults	10	8	5
Hit Ratio	0.17	0.33	0.58

Table.3 Analysis of LRU page replacement algorithm

It is found that page fault decreases when frame size increases and hit ratio is nearly doubled with every increase in frame size. This algorithm is implementable with the fact that it requires a considerable amount of overhead to keep track the time of the pages recently used.

III. RESULT ANALYSIS

After implementation of above three algorithms, consolidated result of page fault and hit ratio is shown in the table below for a comparative study.

	FIFO		Optimal		LRU	
	Page Fault	Hit ratio	Page Fault	Hit ratio	Page Fault	Hit ratio
Frame size 3	9	0.25	7	0.42	10	0.17
Frame size 4	10	0.17	6	0.5	8	0.33
Frame size 5	5	0.58	5	0.58	5	0.58

Table.4 Comparative analysis of Page fault and hit ratio for three algorithms

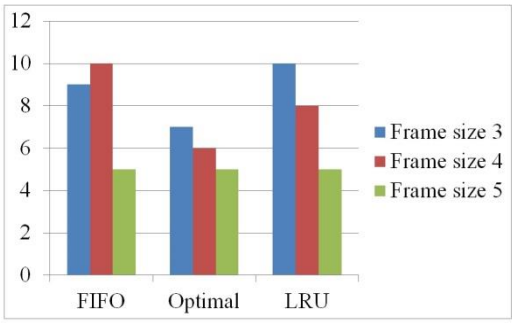

Diagram.1 Page fault comparision of three algorithms

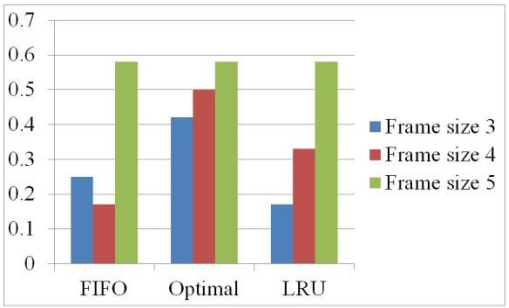

Diagram.2 Hit Ratio comparision of three algorithms

Average page fault for FIFO, Optimal and LRU page replacement algorithm are 8, 6 and 7.3 respectively, also average hit ratio are 0.33, 0.5 and 0.36 respectively.

It indicates that FIFO page replacement algorithm shows the worst performance and Optimal page replacement algorithm shows the best result. FIFO sometimes exhibits inconsistency with the increase in frame size. In any frame size, optimal page replacement algorithm achieves the highest hit ratio followed by LRU page replacement algorithm.

IV. CONCLUSION

Analysis of three page replacement algorithms shows that FIFO is easy to implement, but its performance is worst, Optimal is highly efficient, but very complex to implement. LRU is efficient to a high degree, also implementable. Although three page replacement algorithms are discussed in this paper, there are several algorithms than can be implemented. Adaptive Replacement Cache (ARC), Aging, Clock with Adaptive Replacement (CAR), Not Recently Used (NRU), Not Frequently Used (NFU) are some of the modern perspectives of page replacement techniques. However, its main objective is to construct an efficient, less time consuming technique which can result in minimum page fault. With the up-gradation in modern technology of computation, an enhanced version of these algorithms is also needed to meet its objective. Page replacement algorithm contributes only a fraction in the whole performance of an operating system. However, in a real world computation, any improvement in a small approach also increases the performance of the whole system.

FUTURE WORK

The Optimal is considered to be the best page replacement algorithm, as it generates the lowest page fault rate of all the algorithms. But this algorithm is very difficult to implement because of work load needed to predict the pattern of the future references. In future work, we will try to find out a better way to implement this algorithm by computing the probability of occurrence of future references based on past activities.

REFERENCES

1. A. Silberschatz, P.B. Galvin, G. Gange, Operating System Concepts. Wiley India Edition, 8Th Edition,2010, ch. 9.
2. W. Stallings, Operating System Internals Design and Principles, Prentice Hall, 7th Edition, 2012 ch. 8.
3. J. Kumari, S. Kuamr, D. Prasad, “A Comparison of Page Replacement Algorithms: A Survey” IJSER, valume 7, Issue 12, December-2016.
4. G. Rexha, E. Elmazi, I. Tafa, “ A Comparision of Three Page Replacement Algorithms: FIFO, LRU and Optimal” AJIS MCSER Publishing, Rome-Italy.Vol 4. No 2 S2. August 2015.
5. M. Saktheeswari, K. Sridharan, “ A Stdy on Page Replacement Algorithms”, IJTES., Vol 3., No. 2, 2012.
6. “Page Replacement Algorithm” , 2020. Available at https://en.wikipedia.org/wiki/Page_replacement_algorithm.
7. “Belady’s Anomaly”,2020., Available at https://en.wikipedia.org/wiki/B%C3%A9l%C3%A1dy%27s_anomal.
8. “Page Replacement Algorithms in Operating Systems”, 2020., Available at https://www.geeksforgeeks.org/page-replacement-algorithms-in-operating-systems/.
9. “Page Replacement Algorithms”, 2020., available at https://www.javatpoint.com/os-page-replacement-algorithms.
10. “Optima Page Replacement Algorithm”, 2020., Available at https://www.geeksforgeeks.org/optimal-page-replacement-algorithm/?ref=rp.
11. “Program for page replacement Algorithms|Set 2(FIFO)”, 2020. Available at https://www.geeksforgeeks.org/program-page-replacement-algorithms-set-2-fifo/?ref=rp.
12. “ Program for Least Recently Used(LRU)”,2020., Available at https://www.geeksforgeeks.org/program-for-least-recently-used-lru-page-replacement-algorithm/?ref=rp.

AUTHOR PROFILE

Mr. Jogamohan Medak obtained MCA (Master of Computer Application) degree from the Department of Computer Science and Engineering, Tezpur University, India. He has qualified the UGC-NET criteria for Assistant Professor in Computer Science and Application. He has published research articles in peer reviewed journals and participated in various national level Seminars and Workshops. His area of interest includes programming, Web Designing, Operating System and Software Engineering.

Mr. Partha Pratim Gogoi obtained MCA (Master of Computer Application) degree from the Department of Computer Science and Engineering, Tezpur University, India. He has qualified the SLET criteria for Assistant Professor in Computer Science and Application.. He has published research articles in peer reviewed journals. His area of interest includes programming, Web Designing, Operating System and Software Engineering.