FL: A modified and Combined Approach for Page Replacement in Operating Systems

Dr. Piyush O. Arora[1], Bipin B. Jadav[2]
[1]Shri Shambhubhai V. Patel College of Computer Science and Business Management, V.N.S.G.U.
[1]Shri Shambhubhai V. Patel College of Computer Science and Business Management, V.N.S.G.U.

***Abstract* - Operating system executes processes by page replacement, which is one of the basic requirements of any operating system. There has been continuous and tremendous study towards the design of algorithms that have minimum number of page faults. Variety of page replacement algorithms have been proposed and applied in different situations, in which FIFO (first in first out) and LRU (least recently used) are two of the most popular page replacement policies.**

Modern operating systems strive for throughput maximization by reducing page faults. As page fault rate decreases, efficiency of an algorithm increases, due to the fact that operating system will be busy more in execution of processes, rather than doing I/O.

In the given literature, we propose FL approach, the combination of FIFO and LRU, is such a page replacement technique, which has led to equal or a smaller number of page faults than FIFO and LRU alone. Specifically, this algorithm works well in conditions, if a smaller number of memory frames are available. Conducting tests with a different reference strings and different number of frames, the page fault rate is examined.

Index Terms—**Efficiency, FIFO, FL, LRU, Memory frames, Page Faults, Page replacement, Reference strings.**

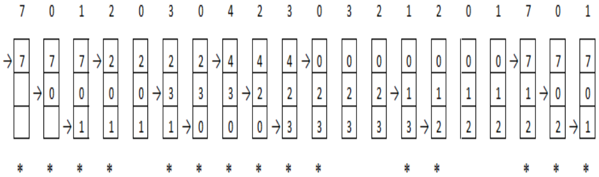
I. INTRODUCTION

Memory management is one of the important tasks of the operating system. Situations may arise, where there are limited memory frames available, to accommodate the pages of processes [1]. It is desirable that process should execute with high throughput in these limited frames [1]. For a processes' page to execute, its valid bit must be set in the page table. There are basic FIFO, Optimal and LRU page replacement policies available to achieve this purpose [2]. These techniques are tested on the grounds of particular reference strings. After simulating the algorithms on the given reference strings, the algorithm that has minimum number of page faults is said to be the best one. As per the research work carried out by [3], LRU is better than FIFO page replacement algorithm. It is generally observed that, as the number of frames increases, page fault rate decreases [4].

FIFO and LRU page replacement algorithms are practically possible, whereas optimal algorithm is not, as it requires the future knowledge of the page, which is to arrive [5].

In the literature [6] a reference string has been used, to test these algorithms. It is:
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

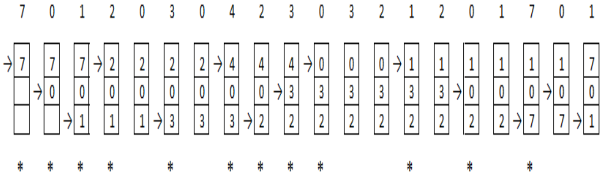
1. FIFO Page Replacement Algorithm



7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
→7	7	7	→2	2	2	2	→4	4	4	→0	0	0	0	0	0	0	→7	7	7
	→0	0	0	0	→3	3	3	→2	2	2	2	2	→1	1	1	1	1	→0	0
		→1	1	1	1	→0	0	0	→3	3	3	3	3	→2	2	2	2	2	→1
*	*	*	*		*	*	*	*	*	*			*	*			*	*	*

Total Page Faults = 15

2. LRU (Least Recently Used) Page Replacement Algorithm



7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
→7	7	7	→2	2	2	2	→4	4	4	→0	0	0	→1	1	1	1	1	1	7
	→0	0	0	0	0	0	0	0	→3	3	3	3	3	3	→0	0	0	→0	0
		→1	1	1	→3	3	3	→2	2	2	2	2	2	2	2	2	→7	7	→1
*	*	*	*		*		*	*	*	*			*		*		*		

Total Page Faults = 12

The above two algorithms as already proved, are useful.

II. MOTIVATION

The objectives behind the design of FL approach are as follows:

- Page fault rate should be as less as possible. FIFO showed 15 and LRU showed 12.

- Once the frames are fully occupied by the pages, it becomes necessary to use the best approach for page replacement.
- Achieve better CPU and memory utilization, thereby increasing throughput of our system.
- Design more sophisticated algorithms by modifying existing algorithms and testing them on different systems.

III. PROPOSED METHODOLOGY

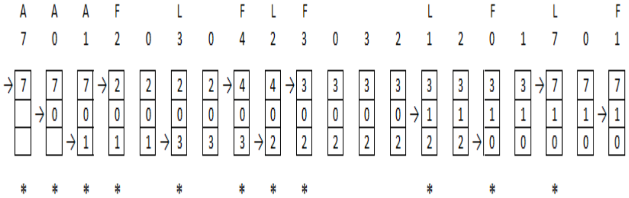
In this paper, the researcher has proposed the combined approach to page replacement. The technique is very simple to understand. It applies FIFO for first page, LRU for the second page, again FIFO for the third page and LRU for the fourth page so on, till the reference string does not end. It means that, the algorithm alternates between the sequences of FIFO and LRU approach.

The researcher has proposed two methods for applying FL approach, namely FL-1 and FL-2. The pseudocode for both the methods is below:

A. *FL-1 TECHNIQUE*

```
Step 1: Input Reference String
Step 2: Initialize variable total_page_faults = 0
Step 3: Repeat steps 4 to 16 while page ≠ NULL
Step 4: Read page from reference string
Step 5:     Check whether frames are empty
            If empty_frames( ) then
Step 6:         Use any frame for current page
                    fill_frames( )
Step 7:     Otherwise, if the frames are not empty
Step 8:         Repeat steps 9 to 15 for FL-1 approach
Step 9:             If valid bit is set
Step 10:                Use the page
Step 11:            Otherwise, proceed to replace page as per
                    FIFO
Replace the page, as per the previous
                      allocation of LRU
Step 12:              total_page_faults = total_page_faults + 1
Step 13:            Read the next page
Step 14:            If valid bit is set
Step 15:                Use the page
Step 16:            Otherwise, proceed to replace page as per
                    LRU
total_page_faults= total_page_faults + 1
                [End of Step 5 if structure]
            [End of Step 3 loop]
Step 17: Output total_page_faults
Step 18: Exit
```

Note: A = any, F = FIFO, L = LRU,
→ = symbol used for FIFO and LRU replacement



Total Page Faults = 11 (less than FIFO and LRU)

The above algorithm is tested and run on the grounds of the same reference string.

It works as follows:

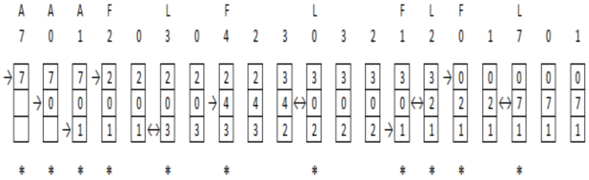
- Case 1, Case 2 and Case 3: Initially, when the frames are empty, it does not matter, which algorithm you use for page replacement. FIFO and LRU both will allocate in the same manner (till frame allocation for page no. 7, 0 and 1), indicated by "A".
- Case 4: Choose FIFO approach, indicated by "→F". As soon as all the frames get occupied and we need to replace an existing page with a new page (replace Page no. 7 by page no. 2).
- Case 5: Page no. 0 already exists. So, no page fault occurs.
- Case 6: Choose LRU approach, indicated by "→L". (Replace page no. 1 by page no. 3).
- Case 7: Page no. 0 already exists. Again, no page fault occurs.
- Case 8: Choose FIFO approach, indicated by "→F". The previous frame allocated by LRU was frame no. 3 and as per FIFO approach, the next frame is frame no. 1 (replace page no. 2 by page no. 4).
- Case 9: Choose LRU approach, indicated by "→L". (Replace page no. 3 by page no. 2).
- Repeat this process again and again, until the reference string completes.

By using this technique, it is observed that, for a given reference string, the page fault rate for the proposed algorithm is less than FIFO and LRU. This proves that FL-1 is more reliable for the given reference string.

B. *FL-2 TECHNIQUE*

```
Step 1: Input Reference String
Step 2: Initialize variable total_page_faults = 0
Step 3: Repeat steps 4 to 16 while page ≠ NULL
Step 4: Read page from reference string
Step 5:    Check whether frames are empty
              If empty_frames( ) then
Step 6:              Use any frame for current page]
                     fill_frames( )
Step 7:          Otherwise, if the frames are not empty
Step 8:              Repeat steps 9 to 15 for FL-1 approach
Step 9:                If valid bit is set
Step 10:                     Use the page
Step 11:               Otherwise, proceed to replace page as
per
                         FIFO
Replace the page, as per the previous
                         allocation of FIFO
Step 12:                total_page_faults = total_page_faults
+ 1
Step 13:                     Read the next page
Step 14:                     If valid bit is set
Step 15:                         Use the page
Step 16:                   Otherwise, proceed to replace page
                     as
                       per LRU
          total_page_faults= total_page_faults + 1
              [End of Step 5 if structure]
          [End of Step 3 loop]
Step 17: Output total_page_faults
Step 18: Exit
```

Note: A = any, F = FIFO, L = LRU,
→ = symbol used for FIFO replacement
↔ = symbol used for LRU replacement



Total Page Faults = 11 (less than FIFO and LRU)

The above algorithm is tested and run on the grounds of the same reference string.

It works as follows:

- Case 1, Case 2 and Case 3: Initially, when the frames are empty, it does not matter, which algorithm you use for page replacement. FIFO and LRU both will allocate in the same manner (till frame allocation for page no. 7, 0 and 1), indicated by "A".
- Case 5: Page no. 0 already exists. So, no page fault occurs.
- Case 6: Choose LRU approach, indicated by "↔L". (Replace page no. 1 by page no. 3).
- Case 7: Page no. 0 already exists. Again, no page fault occurs.
- Case 8: Choose FIFO approach, indicated by "→F". Here, the strategy is to replace the page in next frame using counter variable, maintained by FIFO approach. The previous frame allocated was frame no. 2. So, (Replace page no. 0 by page no. 4).
- Case 9: Choose LRU approach, indicated by "↔L". (Replace page no. 4 by page no. 0).
- Repeat this process again and again, until the reference string completes.

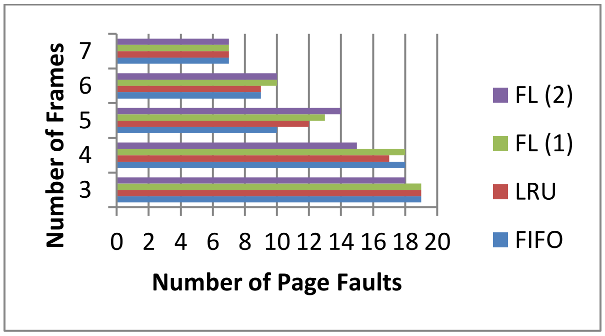
Again, it is observed that, the page fault rate for second technique is also less than FIFO and LRU. This also proves that FL-2 is more reliable for the given reference string.

The researcher has tested the algorithm under various situations and conditions. In some cases, LRU or FIFO shows more faults while in other cases, FL approach shows more page faults. As the prediction of page faults for FIFO and LRU is difficult, so is the case with FL-1 and FL-2 approach. All depends on the reference strings. But, as per the results of simulations performed by testing various reference strings, it was observed that many of them shows less page faults for FL approach.

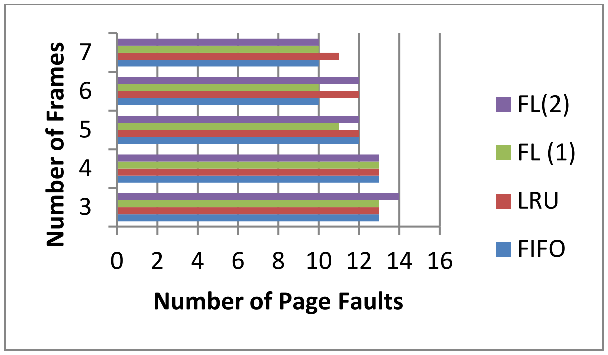
C. *RESULTS AND ANALYSIS*

The researcher performed 90 simulations by assuming 9 different reference strings [6] [7] and 3, 4, 5, 6, and 7 frames. Based on tests performed on different reference strings, following results were observed:

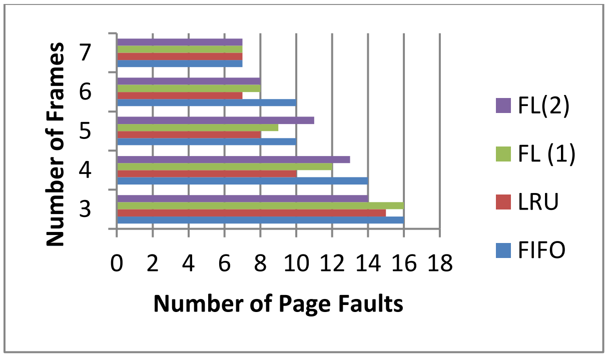
String 1. 1, 2, 3, 4, 1, 6, 5, 6, 2, 1, 3, 7, 4, 2, 1, 3, 5, 7, 2, 1



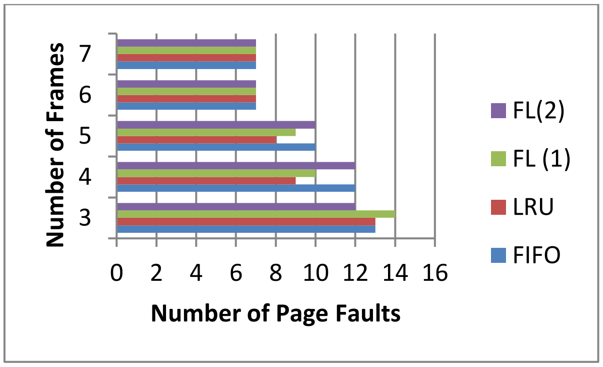
String 2. 1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2



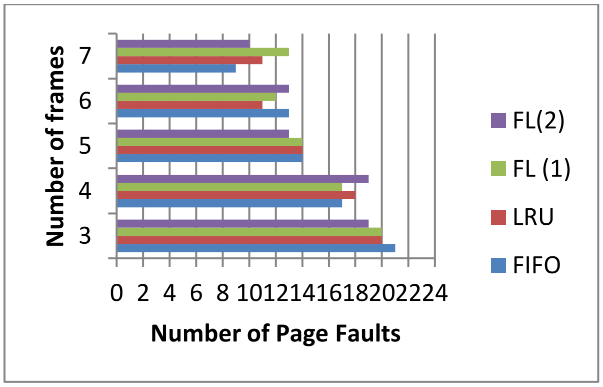
String 3. 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6



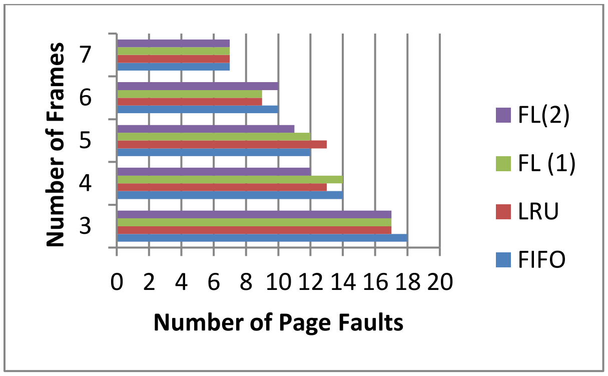
String 4. 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2



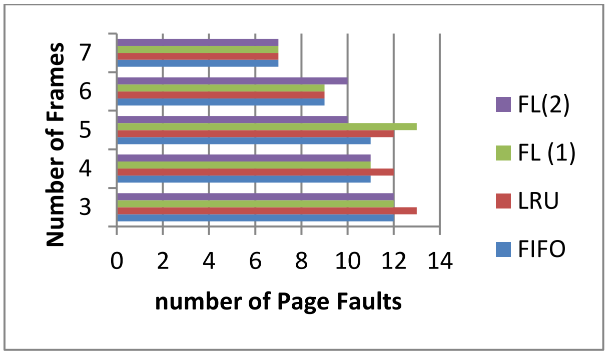
String 5. 1, 0, 2, 2, 1, 7, 6, 7, 0, 1, 2, 0, 3, 0, 4, 5, 1, 5, 2, 4, 5, 6, 7, 6, 7, 2, 4, 2, 7, 3, 3, 2, 3



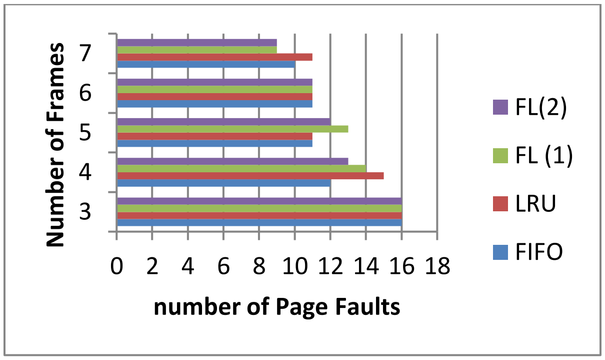
String 6. 1, 2, 3, 4, 1, 4, 5, 6, 2, 3, 7, 1, 2, 4, 6, 2, 1, 3, 2, 6



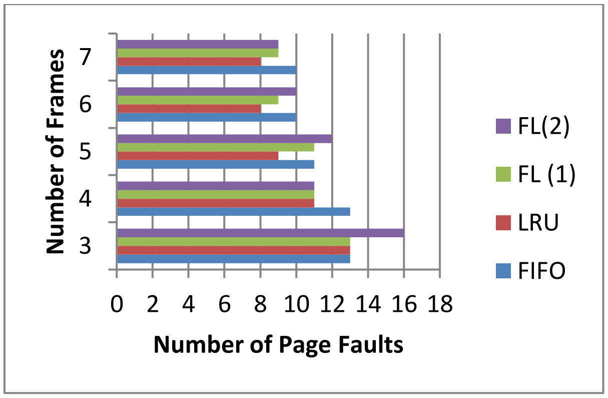
String 7. 1, 3, 4, 4, 3, 2, 1, 7, 5, 6, 4, 2, 1, 2, 4, 3, 2, 1



String 8. 6, 1, 0, 2, 2, 5, 0, 1, 4, 2, 3, 0, 7, 4, 5, 6, 0, 1



String 9. 2, 3, 1, 2, 4, 6, 0, 2, 6, 3, 2, 9, 8, 3, 6, 2, 3, 2



The above results can be summarized from the below given table. The table shows success rates for equal or less page faults.

Herein, a "√" represents that a corresponding event was observed.

Success		Failure		Good		Average	
FL-1	FL-2	FL-1	FL-2	FL-1	FL-2	FL-1	FL-2
√	√	√	√		√	√	
√	√	√	√		√	√	
√	√					√	
√			√			√	
√	√					√	
√	√				√	√	
√	√			√			√
√	√					√	√
√	√				√	√	
√			√		√	√	
√		√		√			
√			√	√			
√	√			√			
	√	√				√	√
√			√		√		
√			√	√			
√	√			√			
√	√					√	√
√	√					√	√
√			√	√	√		
√	√					√	
√			√		√	√	
	√	√		√			
√	√						√
√	√			√			√
√	√				√	√	
√			√		√	√	
√	√			√			
√	√					√	√
√	√				√	√	
	√	√			√	√	
√			√		√		
√	√					√	
√	√					√	√
	√	√				√	√
		√	√		√		
√	√					√	√
√	√			√	√		
√			√			√	
√	√			√			√
√			√			√	
√			√	√			
√			√	√			

The above table summarizes that, out of total 90 simulations performed using FL-1 and FL-2, 66 times the page fault rate is either equal or less than FIFO and LRU, whereas 24 times it is more than FIFO and LRU. It means, FL-1 and FL-2 approach attain 73.34% success rates compared to FIFO and LRU, while the failure rate is 26.66%. The above table also summarizes that, 29 times, the page fault rate is less than FIFO and LRU, while 37 times it is more than FL and LRU. It means, the FL approach outperforms 32.22% times better than FIFO and LRU by giving less page faults. Also, it performs well by giving 41.11% average success rate due equal number of page faults as in FIFO and LRU. Only, in some cases (26.66 %) of the times, the page faults for our algorithm are higher.

During initial stages if the number of frames is large (more free frames list available in the main memory) and the pages in the reference string are unique, FL approach behaves like FIFO. So, the page fault rate is almost equal to that of FIFO. It was specifically observed that algorithm works in a very unique manner, when the number of frames is less. So, the proposed FL approach to page replacement can be used in the situations of memory constraint. Observing the simulations, the problem with FL approach remained the same, as that in FIFO and LRU. Sometimes the victim page replaced by the new page is such that, it just arrived in the memory frame. As per the rules, the recent page should not be removed from the main memory as its execution might be required in the near future. But this condition cannot be avoided, until we take an optimal decision, which is difficult.

IV. CONCLUSION

This paper has two, page replacement policies FL-1 and FL-2 that can be implemented in the operating systems. After performing several simulations, they have been proved to be useful. The page fault rate can be controlled and accordingly thrashing can be avoided, thereby giving high system performance. Future work is open to take optimal decisions about deciding the next allocation for page in the available frames, calculating the time complexity of the algorithms, devising techniques to implement this approach and simulating them on various hardware and software platforms.

V. REFERENCE

[1] Amit S., Kartik R. Nayak, Keval D. Vora, Manish D. Purohit and Pramila M. Chawan, *"A Comparison of Page Replacement Algorithms"*, IACSIT-International Journal of Engineering and Technology, Vol. 3, April 2011.

[2] Genta Rexha, Erand Elmazi, Igli Tafa, *"A Comparison of Three Page Replacement Algorithms: FIFO, LRU and Optimal"*, Academic Journal of Interdisciplinary Studies MCSER, E-ISSN 2281-4612, ISSN 2281-3993, Vol 4 No. 2 S2.

[3] M. Chrobak and J. Noga, *"LRU is better than FIFO*", Algorithmica (1999), Springer-Verlag New York Inc.,

[4] Wesley W. Chu and Holger Opderbeck, *"The page fault frequency replacement algorithm",* University of California, Los Angeles, California.

[5] Pooja Khulbe, Shruti Pant, *"Hybrid Page-Replacement Algorithm",* International Journal of Computer Applications (0975 – 8887), Volume 91 – No. 16, April 2014.

[6] Silberschatz, G. G. (2009). *Operating System Concepts* (8th Edition ed.). R. R. Donnelley/Jefferson, United States of America: John Wiley & Sons. Unc.

[7] Stallings, W. *Operating Systems Internal Design and Principles.*