Indian Journal of Science and Technology, Vol 10(30), DOI: 10.17485/ijst/2017/v10i30/115500, August 2017
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A Novel Longest Distance First Page Replacement Algorithm

Gyanendra Kumar[1]* and Parul Tomar[2]

[1]Department of Information Technology, Raj Kumar Goel Institute of Technology and Management, Ghaziabad - 201001, Uttar Pradesh, India; maurya.gyanendra@gmail.com
[2]Department of Computer Engineering, YMCA University of Science and Technology, Faridabad - 121006, Haryana, India; ptomar_p@hotmail.com

Abstract

Objectives: To improve the performance of computer in program execution by employing Longest Distance First page replacement algorithm in memory management. **Method:** There are many traditional page replacement algorithms used in virtual memory organization like FIFO, LRU, Optimal page replacement, CAR, ARC etc, each of these algorithms tries to reduce the number of page faults in selection of victim page from the memory frames. This paper presents all the popular page replacement algorithms and a new approach named as "Longest Distance First (LDF)" page replacement algorithm. **Findings:** From experimental results and analysis, it has been observed that, LDF produced better performance in terms of page fault rate and implementation overhead than many traditional page replacement algorithms like FIFO, LRU. From the results, the average page fault of LDF is better than FIFO and LRU of taken data set. **Applications:** LDF can be used in virtual memory management to improve performance of computer system by minimize page fault rate.

Keywords: FIFO, LRU, LDF, Memory Management, Optimal Replacement, Virtual Memory

1. Introduction

Computer is an electronic device which executes computer program, and execution of computer program is managed by operating system. Almost every computer consists of operating system which provides all functionality needed in the execution of program. Functionalities of operating system can be resource management, process management, memory management etc. When a program executes, it must be in the main memory of a computer, for that operating system uses different memory management techniques to allocate memory to program. One of the popular memory allocation techniques is demand paging. In paging programs are divided into pages and memory is divided into fixed size frame and frame size must be equal to page size. To allocate memory for execution of program, pages of programs should be loaded in free frames of memory and to keep track a page table is created.

But in virtual memory only few pages of programs are allocated in memory frames to start the execution of programs and all pages are kept in secondary memory of the system. When Central Processing Unit (CPU) references any instruction of page, and if that page is available in main memory frames, then that instruction will be executed by CPU but if the page is unavailable in main memory frames, a page fault will occur. To service page fault, operating system loads needed page from secondary memory to memory frame. If free memory frames are available then needed page will be allocated in any free frame, if free frames are not available then operating system selects a victim page from allocated list. To select victim page from the memory, operating system uses page replacement techniques.

Number of page replacement procedures exists like FIFO[1], RAND[1], LRU[2], Optimal page replacement[1,3], ARC[4], CAR[5], and Aging[6], etc. The main criteria used to

*Author for correspondence

evaluate the replacement algorithms are its page fault rate and overhead to implement it.

Commonly accepted algorithm is LRU because of its performance in term of page fault rate, but it requires high implementation overhead. LDF performs better in terms of page fault rate than LRU over most of the available page reference strings in academic. Its implementation overhead is also less than LRU.

The paper is structured as follows: Section 2 discusses popular existing page replacement algorithms. Section 3 describes the proposed LDF in detail with example illustration. Section 4 describes results and analysis on taken reference strings. Finally, Section 5 concludes the paper.

2. Page Replacement Algorithms

When page fault occurs during the program execution, operating system uses the memory management algorithm to select victim page from primary memory and makes room for required page. Many algorithms have been developed and tested theoretically as well as practically. Some of the popular algorithms are as follows.

2.1 FIFO

It is the simplest page replacement algorithm in implementation but it performs poor in terms of page fault rate. The selection of victim page is based on its arrival in memory. An oldest page is replaced first.

We can explain the working of FIFO with the help of Figure 1 by taking reference string 0 1 2 3 0 1 4 0 1 2 3 4, and three frames in the memory are initially empty.

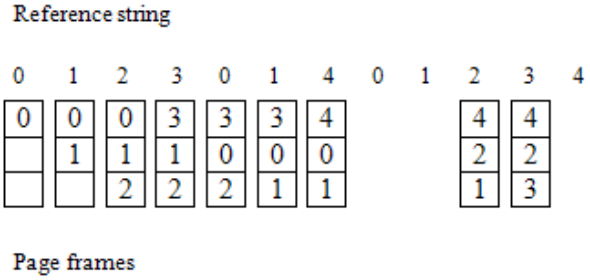

Figure 1. FIFO page-replacement algorithm with 3 memory frames.

In this example total page faults are 9. FIFO algorithm suffers Belady's anomaly[7]: which state that for some page-replacement algorithms the page fault rate may increase as the number of allocated memory frames increase.

2.2 LRU

LRU replacement is based on Locality of reference i.e. it uses the recent past as an approximation of the near future, then we can replace the page that has not been used for the longest period of time[7]. Here the working of LRU can explain with the following example shown in Figure 2. In this example there is a total 10 page fault using LRU.

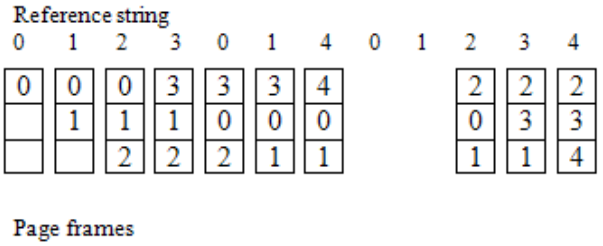

Figure 2. LRU page-replacement algorithm with 3 memory frames.

2.3 Optimal Page Replacement

The basic idea behind this replacement algorithm is:*Replace the page that will not be used for the longest period of time*[7].It can be explained with the below given example shown in Figure 3 and total page faults are 7.

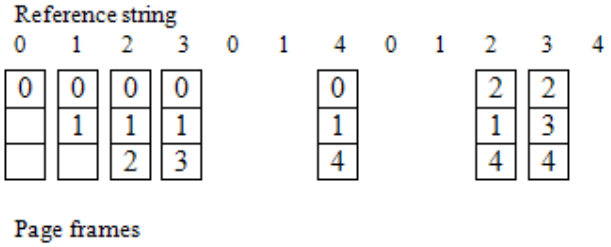

Figure 3. Optimal page-replacement algorithm with 3 memory frames.

Optimal has the lowest page fault rate than all other algorithms, but it is not possible to implement it, because it requires future knowledge. It never suffers Belady's anomaly problem.

There are many other algorithms are available in literature such as ARC, CAR, LIRS[8], CLOCK[9], CLOCK-Pro[9], LRU-K[10] algorithms.

3. Longest Distance First (LDF) Page Replacement Algorithm

The basic idea behind this algorithm is Locality of Reference[11,12] as used in LRU but the difference is that

in LDF, locality is based on distance not on the used references. In the LDF, *replace the page which is on longest distance from the current page.* If two pages are on same distance then the page which is next to current page in anti-clock rotation will get replaced.

Logic behind the LDF is that most of the program or portion of program execute sequentially so if current instruction is n^{th} then probability of executing next instruction closed to it is more than any other instructions. Similarly probability of executing (n+1),(n+2) ... instructions is more than (n-1),(n-2)....instructions respectively. And by considering locality we can say, chances of executing instructions close to current instruction is more than other instructions. This is the main reason behind the LDF. In page replacement if current page is n^{th} then probability of executing next page closed to it is more than other and (n+1), (n+2).. pages probability is more than (n-1),(n-2).. pages.

3.1 Calculation of Distance

For the calculation of distance of a page from the current page, arrange page reference numbers in circular form and count how many number of pages it is away from the current page in both directions, clock wise and anti-clock wise. From these two distances minimum distance will be taken.

For example; suppose a page reference string is 0 1 2 3 0 1 4 0 1 2 3 4. Total pages in this reference string are five i.e., 0 1 2 3 4.Now arrange these numbers in circular form

0
4 1
3 2

Now we can calculate distance of any page from any other page, page number 4 and 1 is on distance of one, and 3 and 2 is on distance of two from page number 0. Similarly distance of page number 2 from 0 is 2 clock wise and 3 anti-clock wise so here the smallest distance will be considered as 2.

3.2 LDF Page Replacement

This sub section describes the working of LDF with the help of following examples.

Example1: Let us consider above page reference string i.e. 0 1 2 3 0 1 4 0 1 2 3 4 and number of frames in memory is 3.

In the Figure 4 given reference string using LDF total page fault is 8. The first three page reference cause page fault because initially memory frames are free and no page replacement is required. Fourth page reference i.e., page number 3 will also cause page fault as now memory frames are full, so it requires page replacement, using LDF, page 1 will be replaced with page 3 because distance of 0 and 1 from 3 is 2 and distance of page 2 from 3 is 1. Pages 0 and 1 are on same distance from current page 3 so the page which is next to page 3 will be replaced in anti-clock rotation and it is page number 1. Fifth page reference 0 will not cause page fault. Sixth page 1 will cause page fault so page 3 will be replaced because it is on longest distance from 1. Seventh page 4 will also cause page fault so page 2 will be replaced with 4. Page references Eight, Nine and Twelfth will not cause page fault but tenth and eleventh page will cause page faults. Total page faults for above taken reference string will be 8.

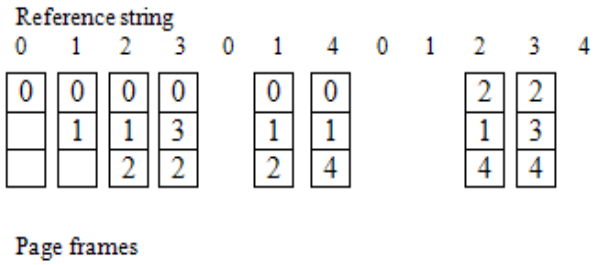

Figure 4. LDF page-replacement algorithm with 3 memory frames.

Example 2: Let us consider page reference string 0 2 1 3 5 4 6 3 7 4 7 3 3 5 5 3 1 1 1 7 1 3 4 1 and frames in memory is four. What is the total page fault?

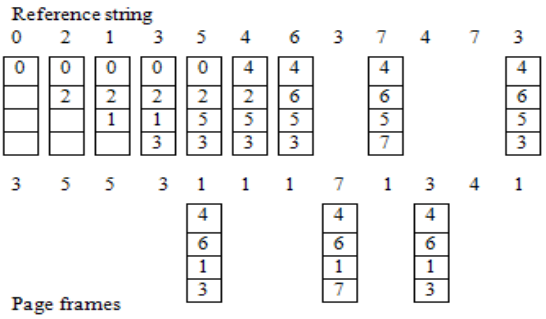

Figure 5. LDF page-replacement algorithm with 4 memory frames.

As shown in Figure 5 total page fault in taken reference string is 12. Starting four pages will cause page fault but no page replacement is required as frames are free. Fifth

page number 5 will also cause page fault which requires page replacement and page number 1 will be replaced because it is on the longest distance from current page 5. Sixth page number 4 will also cause page fault and page number 0 will be replaced as it is on longest distance from current page 4. Seventh page number 6 will also cause page fault and page number 2 will get replaced, and so on all other references will take place.

3.3 Distance Page Fault (Limitation)

LDF suffers a problem named as *Distance Page Fault*, when two pages are on longest distance from each other, they are called *distance page* and if they appear in reference string consecutively then they will replace each other and it will cause page fault. This type of page fault in LDF is known as Distance Page Fault. Let us take reference string 0 1 0 0 2 0 3 3 2 1 1 2 1 3 1 3 1 to illustrate the Distance page fault problem. In this reference string page 3 and 1 are on longest distance from each other and appearing in string consecutively so they will replace each other and cause consecutive page fault.

Chance of occurrence of Distance page fault with a program is less because it will happen with only those programs where two jump statement of different page executes by calling each other and pages must be distance pages.

4. Results and Analysis

This section will give a comparative analysis between FIFO, LRU, Optimal and proposed LDF algorithms.

Result analysis has been done using software written in C. Analysis is based on the following set of reference strings listed in Table 1. These page reference strings are used in academics to analyse number of page faults of these algorithms. Initially program checks page faults of each reference string by considering three memory frames and again it checked page faults by considering four memory frames.

From the results it has been clearly shown that the performance of LDF is better than the FIFO and LRU and less than Optimal algorithm in terms of page faults. Implementation overhead of LDF is less than FIFO, LRU and Optimal because it does not require any extra hardware or support to implement it. From experiments on different page reference strings it is observed that LDF does not suffer Belady's anomaly problem but it needs extra and real experiments to ensure it. Details of the experiment and results are shown in the given tables and graphs.

Table 1 list out the page reference strings used for the purpose of evaluation of performance of algorithms in terms of page faults.

Table 2 shows the page faults with corresponding algorithms listed, these algorithms have been checked by considering three and four frames as shown in Table 2. From Table 2 it is clear that total and average number of page faults occurrence in LDF on taken data is less than LRU and FIFO.

Figure 6 shows the comparison graph between FIFO, LRU, LDF and Optimal page replacement algorithms when three memory

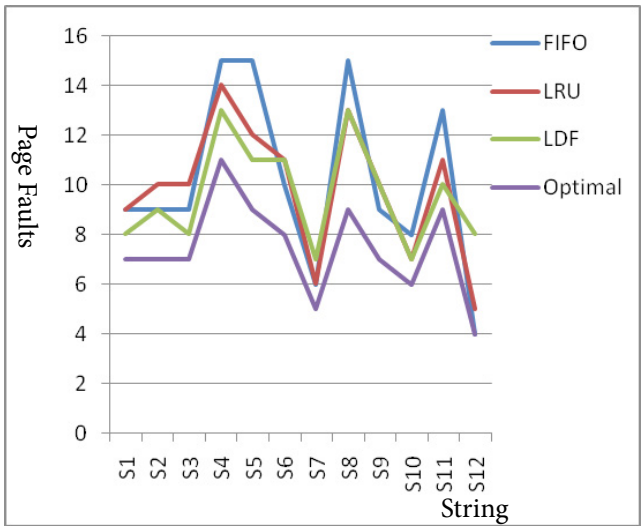

Figure 6. Variation of page fault in FIFO, LRU, LDF and Optimal when three memory frames taken.

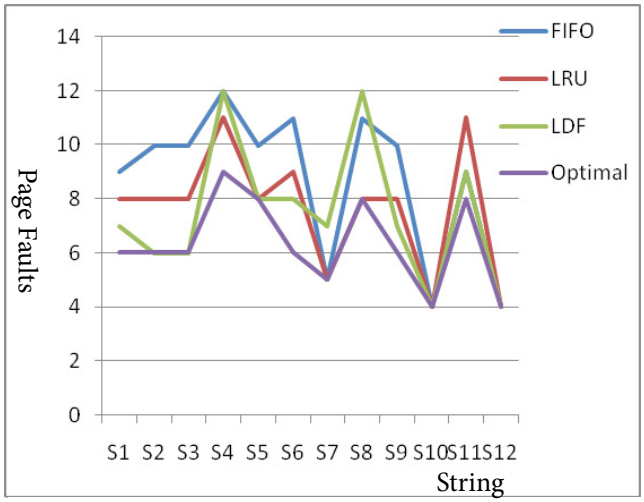

Figure 7. Variation of page fault in FIFO, LRU, LDF and Optimal when Four memory frames taken.

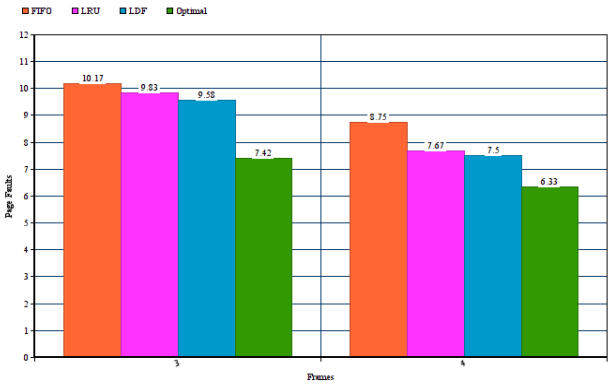

Figure 8. Average page faults of taken data set of reference strings.

frames considered. It is clear from this graph that LDF outperform FIFO and LRU in terms of page faults.

Figure 7 shown the comparison graph between FIFO, LRU, LDF and Optimal page replacement algorithms when four memory frames considered. In this case LDF again outperform FIFO and LRU in terms of page faults.

Figure 8 shows the comparison bar chart of average page faults of all page reference strings listed in table 1. LDF's average page fault is 9.58 and 7.5 when three and four memory frames considered respectively, it is less than FIFO and LRU.

Table 1. List of page reference strings

String No.	Page Reference String
S1	0 2 1 6 4 0 1 0 3 1 2 1
S2	1 2 3 4 1 2 5 1 2 3 4 5
S3	0 1 2 3 0 1 4 0 1 2 3 4
S4	0 2 1 3 5 4 6 3 7 4 7 3 3 5 5 3 1 1 1 7 1 3 4 1
S5	7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
S6	5 4 3 2 1 4 3 5 4 3 2 1 5
S7	4 7 0 7 1 0 1 2 1 2 7 1 2
S8	5 0 2 1 0 3 0 2 4 3 0 3 2 1 3 0 1 5
S9	4 3 2 1 4 3 5 4 3 2 1 5
S10	0 2 1 0 3 0 2 3 0 3 2 1 3 0 1
S11	1 0 5 1 1 3 5 1 5 3 4 5 2 1 3 0 1 4 0 5
S12	0 1 0 0 2 0 3 3 2 1 1 2 2 3 2 1 3

In strings S7 and S8, LDF suffers from distance page fault problem because two distance pages are present one after the other.

Table 2. Comparison of page fault between LDF, FIFO, LRU and optimal

String ⇩	FIFO		LRU		LDF		Optimal	
Frames ⇨	3	4	3	4	3	4	3	4
S1	9	9	9	8	8	7	7	6
S2	9	10	10	8	9	6	7	6
S3	9	10	10	8	8	6	7	6
S4	15	12	14	11	13	12	11	9
S5	15	10	12	8	11	8	9	8
S6	10	11	11	9	11	8	8	6
S7	6	5	6	5	7	6	5	5
S8	15	11	13	8	13	12	9	8
S9	9	10	10	8	10	7	7	6
S10	8	4	7	4	7	4	6	4
S11	13	9	11	11	10	9	9	8
S12	4	4	5	4	8	4	4	4
Total	122	105	118	92	**115**	**90**	89	76
Average	10.17	8.75	9.83	7.67	**9.58**	**7.5**	7.42	6.33

5. Conclusion and Future Scope

This paper gives a comparative study of commonly used page replacement algorithms such as FIFO, LRU and Optimal. A new approach for page replacement named "LDF" has been proposed and compared with existing algorithms. This technique makes use of distance from current page. From the observation it has been found that LDF has better performance as compare to FIFO and LRU but lower performance as compare to Optimal. It has also been observed analytically that implementation overhead of LDF is less than others.

LDF is tested against the page reference strings used in academics but it needs to be tested it in real situation of paging. The distance page fault problem of LDF has already been mentioned. In future, researcher can address distance page fault problem and perform rigorous testing in real scenario to make improvements.

6. References

1. Belady LA. A study of replacement algorithms for virtual storage computers. IBM Systems Journal. 1966; 5(2):78–101. Crossref
2. Mattson RL, Gecsei J, Slutz DR, Traiger IL. Evaluation techniques for storage hierarchies. IBM System Journal. 1970; 9(2):78–117. Crossref
3. Aho AV, Denning PJ, Ullman JD. Principles of optimal page replacement. Journal of the ACM. 1971; 18(1):80–93. Crossref
4. Meigiddo N, Modha DS. ARC a self-tuning low overhead replacement cache. IEEE Transactions on Computers. 2003; 115–30.
5. Bansal S, Modha DS. CAR clock with adaptive replacement. FAST 04 Proceedings of the 3rd USENIX Conference on File and Storage Technologies; 2004. p. 187–200.
6. Tanenbaum AS. Modern operating systems. 4th ed. Pearson; 2015.
7. Gagne G. Silberschatz A, Galvin PB. Operating systems concepts. 7th ed. 2005.
8. Jiang S, Zhang X. LIRS an efficient policy to improve buffer cache performance. IEEE Transactions on Computers. 2002 Jun; 30(1):31–42.
9. Jiang S, Zhang X, Chen F. CLOCK-pro an effective improvement of the CLOCK replacement. ATEC 05 Proceedings of the Annual conference on USENIX Annual Technical Conference; 2005. p. 35–5.
10. Oneil EJ, Oneil EP, Weikum G. An optimality proof of the LRU-K Page replacement algorithm. Journal of the ACM. 1999 Jan; 46(1):92–112. Crossref
11. Denning PJ, Kahn KC. A study of program locality and lifetime functions. ACM SIGOPS Operating Systems Review. 1975 Nov; 9(5):207–16. Crossref
12. Denning, PJ. The working set model of program behavior. Communications of the ACM. 1968 May; 11(5):323–33. Crossref