INTERNATIONAL JOURNAL OF ADVANCE RESEARCH AND DEVELOPMENT

IJARnD

(Volume3, Issue7)

Available online at: www.ijarnd.com

Analysis of page replacement algorithms using C++

Aye Aye Cho

Lecturer, University of Computer Studies, Yangon, Myanmar

ABSTRACT

Page replacement algorithms are important for virtual memory management and it helps the operating system to decide which memory page can be moved out making space for the currently needed page. Whenever a process refers to a page that is not present in memory, a page fault occurs. Each algorithm has the objective to minimize the number of page faults. With minimum page faults, the performance of the process is increased. The purpose of this paper is to analyze the three algorithms via. First in First out (FIFO), Least Recently Used (LRU) and Optimal Page Replacement (OPT) is implemented using C++ programming. We will present the implementation of three algorithms and compare their performance on generated virtual traces.

Keywords— *Page fault, Page replacement algorithms*

1. INTRODUCTION

Page replacement is basic to demand to the page. It completes the separation between logical memory and physical memory. With this mechanism, an enormous virtual memory can be provided for programmers on a smaller physical memory. With non - demand paging, user addresses are mapped into physical addresses, so the two sets of addresses can be different. All the pages of a process still must be in physical memory, however. With demand paging, the size of the logical address space is no longer constrained by physical memory. If we have a user process of 20 pages, we can execute it in ten frames simply by using demand paging and using a replacement algorithm to find a free frame whenever necessary. If a page that has been modified is to be replaced, its contents are copied to the disk. A later reference to that page will cause a page fault. At that time, the page will be brought back into memory, perhaps replacing some other page in the process. If we have multiple processes in memory, we must decide how many frames to allocate to each process. Father, when page replacement is required, we must select the frames that are to be replaced. Designing appropriate algorithms to solve these problems is an important task because disk I/O is so expensive. Even slight improvements in demand-paging methods yield large gains in system performance. There are many different page-replacement algorithms. In general, we want the one with the lowest page-fault rate. Evaluate an algorithm by running it on a particular string of memory references and computing the number of page faults. The string of memory references is called a reference string. We can generate reference strings artificially or we can trace a given system and record the address of each memory reference. The page replacement concept can be used in many areas of computer design.

2. ANALYSIS OF PAGE REPLACEMENT ALGORITHMS

There are many different page-replacement algorithms. Every operating system probably has its own replacement scheme. How do we select a particular replacement algorithm? In general, we want the one with the lowest page-fault rate. Evaluate an algorithm by running it on a particular string of memory references and computing the number of page faults. The string of memory references is called a reference string. We can generate reference strings artificially or we can trace a given system and record the address of each memory reference. Page replacement takes the following approach. If no frame is free, we find one that is not currently being used and free it. We can free a frame by writing its contents to swap space, and change the page table to indicate that the page is no longer in memory. We can now use the freed frame to hold the page for which the process faulted.

2.1 FIFO page replacement

The simplest page-replacement algorithm is a FIFO algorithm. A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. It is not strictly necessary to record the time when a page is brought in. We can create a FIFO queue to hold all pages in memory. We replace the page at the head of the queue. When a page is brought into memory, we insert it at the tail of the queue.

Let us consider a 20 pages common reference string 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6 for two, three and four frames in memory to find the page faults using FIFO page replacement algorithm.

Reference strings:

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For two frames:

1	1	3	3	2	2	5	5	2	2	2	3	3	6	6	2	2	2	3	3
	2	2	4	4	1	1	6	6	1	1	1	7	7	3	3	1	1	1	6

Number of page faults:

F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F

In the above reference string with 2-page frames, the number of page faults is 18.

Reference strings:

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For two frames:

1	1	1	4	4	4	4	6	6	6	6	3	3	3	3	2	2	2	2	6
	2	2	2	2	1	1	1	2	2	2	2	7	7	7	7	1	1	1	1
		3	3	3	3	5	5	5	1	1	1	1	6	6	6	6	6	3	3

Number of page faults:

F	F	F	F		F	F	F	F	F		F	F	F	F	F	F		F	F

In the above reference string with 3-page frames, the number of page faults is 16.

Reference strings:

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For Four frames:

1	1	1	1	1	1	5	5	5	5	5	3	3	3	3	3	1	1	1	1
	2	2	2	2	2	2	6	6	6	6	6	7	7	7	7	7	7	3	3
		3	3	3	3	3	3	2	2	2	2	2	6	6	6	6	6	6	6
			4	4	4	4	4	4	1	1	1	1	1	2	2	2	2	2	

A number of page faults:

F	F	F	F			F	F	F	F		F	F	F	F	F	F		F	F

The above reference string with the 4-page frame, the number of page faults is 14.

The FIFO page-replacement algorithm is easy to understand and program. However, its performance is not always good. Belady's anomaly is an unexpected result in FIFO page replacement algorithm. For some page - replacement algorithms, the page fault rate may increase as the number of allocated found expect that giving more memory to a process would improve its performance. In some early research, .investigators noticed that this assumption was not always true. Belady's anomaly was discovered as a result.

2.2 Optimal page replacement

One result of the discovery of Belady's anomaly was the search for an optimal page-replacement algorithm. An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms, and will never suffer from Belady's anomaly. Such an algorithm does exist and has been called OPT or MIN. It simply replaces the page that will not be used for the longest period of time.

Reference strings:

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For two page frames :

1	1	3	4	4	1	5	6	6	1	1	3	3	3	3	3	1	1	1	6
	2	2	2	2	2	2	2	2	2	2	2	7	6	6	2	2	2	3	3

A number of faults:

F	F	F	F		F	F	F		F		F	F	F		F	F		F	F

The above reference string with the two-page frame, the number of page fault is 15.

Reference strings:

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For Three page frames:

1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	3
	2	2	2	2	2	2	2	2	2	2	2	7	7	7	2	2	2	2	2
		3	4	4	4	5	6	6	6	6	6	6	6	6	6	1	1	1	6

Number of faults:

F	F	F	F			F	F				F	F			F	F			F

The above reference string with the three-page frame, the number of page faults is 11.

Reference strings:

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For Four page frames:

1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	7		1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			4	4	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6

Number of faults:

F	F	F	F			F	F					F				F			

The above reference string with four frames, the number of page fault is 8.

2.3 LRU page replacement

If the optimal algorithm is not feasible, perhaps an approximation to the optimal algorithm is possible. The key distinction between the FIFO and OPT algorithms is that the FIFO algorithm uses the time when a page was brought into memory, the OPT algorithm uses the time when a page is to be used. If we use the recent past as an approximation of the new future, then we will replace the page that has not used for the longest period of time. This approach is the least-recently-used algorithm.

Reference strings:

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For two page frames :

1	1	3	3	2	2	5	5	2	2	2	2	7	7	3	3	1	1	3	3
	2	2	4	4	1	1	6	6	1	1	3	3	6	6	2	2	2	2	6

Number of faults:

F	F	F	F	F	F	F	F	F	F		F	F	F	F	F	F		F	F

The above reference string with two frames, the number of page fault is 18.

Reference strings:

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For Three page frames:

1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2
	2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3	3
		3	3	3	1	1	1	2	2	2	2	2	6	6	6	1	1	1	6

Number of faults:

F	F	F	F		F	F	F	F	F		F	F	F		F	F			F

The above reference string with three frames, the number of page fault is 15.

Reference strings:

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For Four page frames:

1	1	1	1	1	1	1	1	1	1	1	1	7	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	5	5	5	5	5	3	3	3	3	3	3	3	3	3
			4	4	4	4	6	6	6	6	6	7	7	7	7	1	1	1	1

Number of faults:

F	F	F	F			F	F				F	F	F			F			

The above reference string with four frames, the number of page fault is 10.

2.4 Implementation of page replacement algorithms in C++

The number of page faults of three algorithms via. FIFO LRU and OPT are calculated by entering the same number of pages, the same sequence and the same page frame in main memory. The purpose is to test the performance of the three algorithms with the same sequence. The number of page faults can be calculated for all the three algorithms. Window 7 Ultimate operating system is chosen, C++ is used to compiled and execute the application. Trace data used is very simple and the results based on it can't be used as a real-world performance indicator.

Page fault charts give a graphical overview of how the algorithm behaves compared to the optimal algorithm.

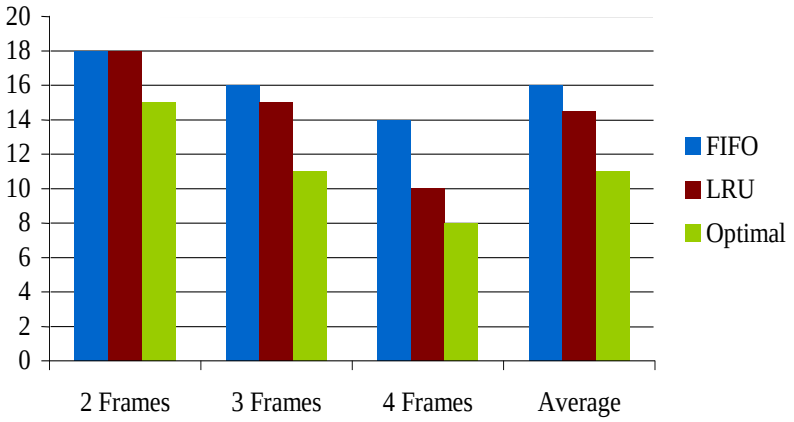

Fig. 1: Graph showing page faults with 2, 3 and 4 frames and they're average

Table 1: Page faults for frame size 2, 3 and 4

Algorithm	FIFO	LRU	Optimal
Page Frames(2)	18	18	15
Page Frames(3)	16	15	11
Page Frame(4)	14	10	8
Average	16	14.3	11.3

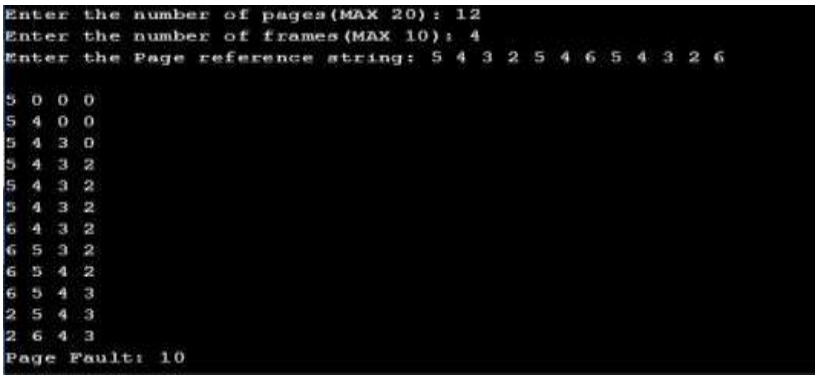

Fig. 2: Output of First in First out page replacement program

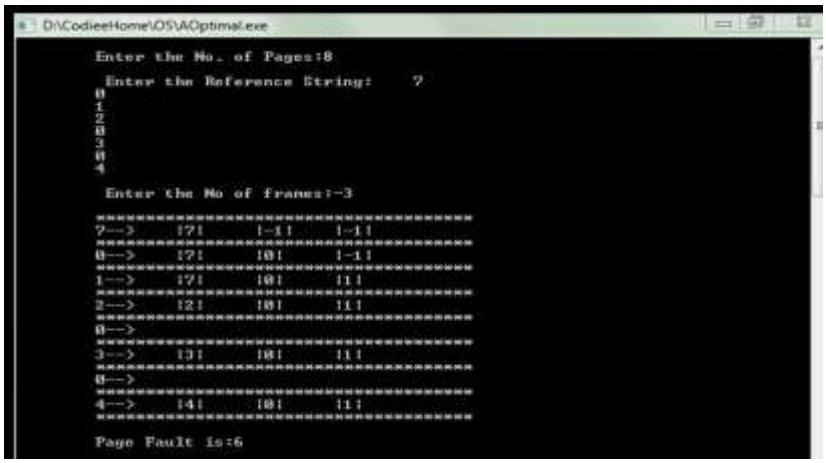

Fig. 3: The output of the optimal page replacement program

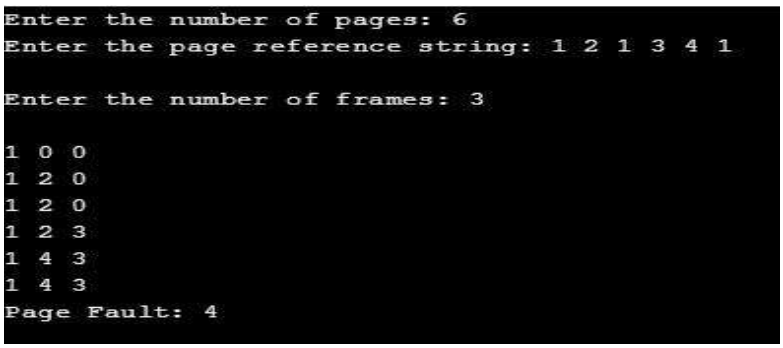

Fig. 4: Output of LRU page replacement Program

3. CONCLUSION

In the above methods or algorithms .we have found that optimal page replacement algorithm results in the best algorithm because the average page faults in all three cases with page frame size 2,3 and 4 is less as compared to FIFO and LRU. Implementation of these algorithms is done as an offline replacement with demand paging policy. In this analysis, we focus on the offline performance and thus the overhead of the algorithms is not analyzed.

4. ACKNOWLEDGMENT

I would like to thank my colleagues, M.Sc. (Nuclear Physics) at the University of Computer Studies (Maubin) for their participation, discussions and helpful support.

5. REFERENCES

[1] A.Silberschatz, P.Galvin, G.Gogne, "Operating Systems Concepts "6th ed, Danvers, Mass: John Wiley and sons, 2003
[2] Heikki Paajanen "Page replacement in operating system memory management" Master's thesis, University of Jyvaskyla, 2007
[3] Dr.C.Tayer, Sumit Sehgal" Research paper on virtual memory" online accessed http:/www.danepraine.com
[4] Wikipedia." Page fault-Wikipedia, the free encyclopedia "2018[online: accessed 2018 june]
[5] Sourabh Shastri, Anand Sharma, Vibhaker Mansotru" The international journal of engineering and science(IJES)5(1)-2016.53-57
[6] http://tutorialheap.com

BIOGRAPHY

Daw Aye Aye Cho
Lecturer
Faculty of computer science
Universities of computer studies
Yangon

