E-ISSN 2281-4612
ISSN 2281-3993

Academic Journal of Interdisciplinary Studies
MCSER Publishing, Rome-Italy

Vol 4 No 2 S2
August 2015

A Comparison of Three Page Replacement Algorithms: FIFO, LRU and Optimal

Genta Rexha[1]

Erand Elmazi[2]

Igli Tafa[2]

[1]University of Elbasan, Faculty of Economy, Department of Marketing and Engineering
[2]Polytechnic University of Tirana, Faculty of Information Technology, Department of Computer Engineering
Email: gentarexha@yahoo.com

Doi:10.5901/ajis.2015.v4n2s2p56

Abstract

The speed at which the process will be executed doesn't depend only on the implementation of the architecture of the computer, the frequency of the clock, but also regard to the politics the algorithm follows and the data structure the algorithm is implemented. The usage of efficient page replacement algorithms, that choose which page in memory will be swapped if a page fault, are required to implement in a virtual memory system. A lot of algorithms are developed during the years for page replacement. Each algorithm has the aim to have less number of page faults. With less page faults we have an improvement in performance and the speed of the process is increased. In this paper three algorithms (FIFO, LRU and Optimal page replacement algorithms) will be tested and will be shown the one that has the best performance.

Keywords: *memory management, replacement policy, page replacement algorithm*

1. Introduction

In the memory management system the replacement of the pages is a main concept. When the Kernel creates the process with system calls (Tanenbaum 2008, Soares et al. 2010), pages is required to be in the main memory. These pages placed in the memory will be required by the process (Klues et al. 2010). If we will not find the pages in the main memory, we will face with a situation that is called page fault. When a page fault occurs, the process needs to find the page that caused the page fault in disk. If there is a free space in main memory it will put in this area. If there isn't free space it is necessary to make a page replacement with the algorithm that is implemented. After that the page that is removed will be written in disk. Then the process can continue the job, because the required page is in the main memory and the process can read it.

There are many page replacement algorithms (Tanenbaum 2008) such as: The Optimal, The Not Recently Used (NRU), The Not Frequently Used (NFU), The First-in-First-out (FIFO), The Second Chance, The Clock, The Least Recently Used (LRU), Aging, The Working Set, The Working Set Clock that are available in memory management (Windows Internal Book).

The page replacement concept can be used in many areas of computer design. Two of most interesting and important uses are*:* cache (Brehob et al. 2004) and web servers.

2. Related Works

There are a lot of works that regards page replacement algorithms (Comen et al. 2009). Many papers regard LRU page replacement algorithm. Here, we focus on the work of Heikki Paajanen (Paajanen 2007) about comparison of page replacement algorithms (Chavan et al. 2011). In this paper, the author makes a description of a memory management and describes politics of all page replacement algorithms that are used nowadays. He proposes some formal models to be used as theoretical analyses. He, also, makes a comparison of page hit ratio. Another interesting work is CFLRU that is an improvement of LRU for flash memory (Park et al. 2006). In this paper, the proposed algorithm takes into the consideration the imbalance of read and writes costs of flash memory when replacing pages. Clean-First LRU keeps a certain amount of dirty pages with purpose in page cache to reduce the number of flash write operations. This reduces the average replacement cost by 28.4% in swap system.

3. Algorithms description

In this section three algorithms (FIFO, LRU and OPTIMAL) are presented. The results show when a page fault happened and how to replace (swapped) a page frame in the memory with another page frame that is in disk.

3.1 The First-in-First-out Algorithm (FIFO)

In the first step, the pages are loaded in the main memory. If the page is in the memory, we pass in the other page and *n* is increased by one (Figure 1).

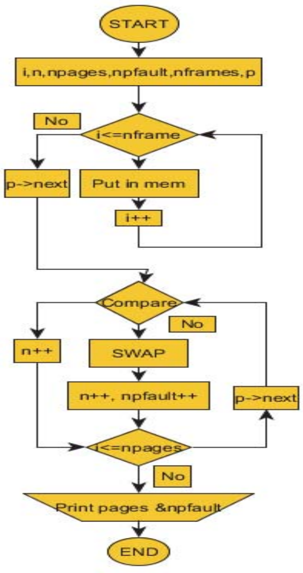


Figure 1: FIFO flowchart diagram

When a process requires a page that isn't in the memory, a page fault occurs. A page replacement needs to be in action. *Which page chooses FIFO (Lo et al 2001) to replace?* FIFO selects the page that is entered the first in main memory. FIFO removes the page that is older.

In Table 1 is shown how FIFO works. A sequence with 8 number of pages and a page frame with size = 3 is chosen.

Table 1: FIFO page replacement algorithm

Sequence:

2	3	4	2	1	3	7	5

Steps	1	2	3	4	5	6	7	8	9
Pages	2	2	2	2	1	1	1	1	1
	-	3	3	3	3	3	7	7	7
	-	-	4	4	4	4	4	5	5

In the fourth step doesn't happen any replacement, because the page is in the memory. In the fifth step, page 2 that was placed the first is replaced by page 1.

3.2 The Optimal Algorithm

In this section the Optimal algorithm is introduced. At the first step, pages are loaded from the disk to main memory. *How many page frames?!* The number of page frames is defined from the capacity of memory and they are loaded to invoke memory until it is full. When a page fault occurs, a page needs to be swapped (Figure 2). The operating system swaps out the page, whose next use will occur farthest in the future.

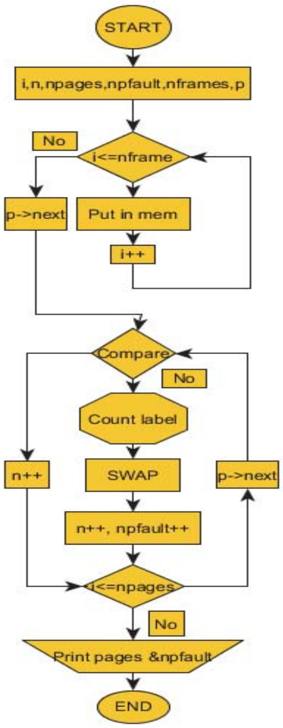


Figure 2: Optimal flowchart diagram

For example, a page that is not going to be used for the next 2 seconds will be swapped out over a page that is not going to be used for the next 0.8 seconds.

Table 2 shows how the Optimal algorithm works. The same sequence with 8 number of pages and a page frame with size = 3 is chosen.

Table 2: The Optimal page replacement

Sequence:

2	3	4	2	1	3	7	5

Steps	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Pages	**2**	**2**	**2**	**2**	**1**	**1**	**7**	**5**
	-	**3**	**3**	**3**	**3**	**3**	**3**	**3**
	-	**-**	**4**	**4**	**4**	**4**	**4**	**4**

In the fourth step the page 2 is in the memory. In the fifth step the page 2 is replaced with page 1.

3.3 *LRU page replacement algorithm*

This algorithm is based on the probability theory or in a logic idea. This idea consists in pages, which have been heavily used in the last few instructions, which will probably be used again in the next few. Pages that aren't used for a long time may stay in this state again. This idea suggests the philosophy of this algorithm, when a page fault occurs throw out (swapped) the page, that isn't used for a long time. Page that is swapped from memory is the page with small counter. This is shown in the flowchart diagram in Figure 3. The page, which instructions haven't been executed for a long time, is replaced. To implement LRU (Kavar et al. 2013) is chosen a linked list to keep all pages in memory, with most recently used page at the front and least recently used page at the tail.

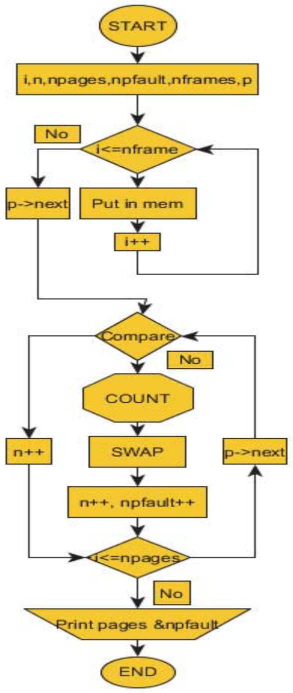


Figure 3: LRU flowchart diagram

Table 3 shows how the LRU algorithm works. The same sequence with 8 number of pages and a page frame with size = 3 is chosen.

Table 3: LRU page replacement

Sequence:

2	3	4	2	1	3	7	5

Steps	1	2	3	4	5	6	7	8
Pages	2	2	2	2	2	2	7	7
	-	3	3	3	1	1	1	5
	-	-	4	4	4	3	3	3

In the fourth step doesn't happen any replacement because the page is in the memory. In the fifth step page 3, that is used at least, is replaced by page 1.

4. Results

The number of page faults of three algorithms is tested (FIFO, LRU and Optimal) by entering the same number of pages, the same sequence and the same page frame in main memory.

4.1 Environment

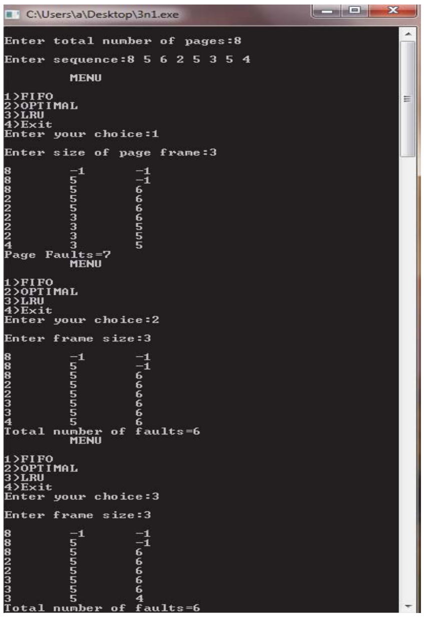
Windows 7 Ultimate operating system is chosen. Dev C++ is used to compile and execute the application.

4.2 Test phase

A random sequence of numbers is chosen, because the application generates different sequences, and the purpose is to test the performance of the three algorithms with the same sequence. Thus, the following sequence is chosen:

85625354235326256856234213754315

and it is used for respectively 8 pages, 16 pages, 24 pages and 32 pages.



```
Enter total number of pages:8

Enter sequence:8 5 6 2 5 3 5 4

        MENU

1>FIFO
2>OPTIMAL
3>LRU
4>Exit
Enter your choice:1

Enter size of page frame:3

8       -1      -1
8       5       -1
8       5       6
2       5       6
2       5       6
2       3       6
2       3       5
2       3       5
4       3       5
Page Faults=7
        MENU

1>FIFO
2>OPTIMAL
3>LRU
4>Exit
Enter your choice:2

Enter frame size:3

8       -1      -1
8       5       -1
8       5       6
2       5       6
2       5       6
3       5       6
3       5       6
4       5       6
Total number of faults=6
        MENU

1>FIFO
2>OPTIMAL
3>LRU
4>Exit
Enter your choice:3

Enter frame size:3

8       -1      -1
8       5       -1
8       5       6
2       5       6
2       5       6
3       5       6
3       5       6
3       5       4
Total number of faults=6
```

Figure 5: Screenshot of the application

In the first step the program requires the number of pages that are pretended to be in main memory. After that, it requires to put the sequence of the pages. In the next step, a menu is printed. A choice can be make. The number of frames is entered, that is the capacity of the main memory. The application shows for each step the pages that are currently in main memory and the number of page faults. A table with number of page faults, for each algorithm, with page frame size = 3 and with respectively 8 pages, 16 pages, 24 pages and 32 pages is generated.

Table 4: Number of page faults

	Page frame size = 3			
No. pages / Algorithm	8	16	24	32
FIFO	7	12	18	25
Optimal	6	8	12	18
LRU	6	9	14	20

The results are also given in Figure 5.

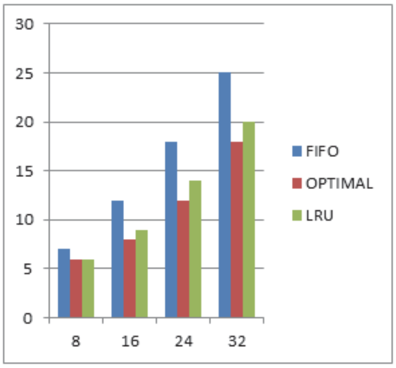


Figure 6: Results of the algorithms

5. Conclusions

The purpose of this work was to define the algorithm that realizes the best performance of the system. A good page replacement algorithm can reduce the page faults, when the program is executing, reduce the number of I/O, and then increase the system's efficiency effectively. The time is a critical point for the systems. An improvement in the performance of the system, having less page faults is made. Optimal results the best algorithm (Figure 5). FIFO has the worst performance. It has more page faults (*degenerates*) when the number of pages is increased. Many tests show, that FIFO, sometimes, makes a wrong decision. It deletes a page from memory and brings it back only after two steps. This takes many times, because it writes a page in disk and brings it back in main memory in two steps. LRU is the better algorithm to implement in these conditions. It is clearly shown in Figure 5. The number of page faults for LRU is near to Optimal page replacement. The results are clearer when the number of pages increases. There are three problems to face with. The first is to generate automatically the same sequence for three algorithms. It is impossible, because it generates three different sequences. The second problem is to implement properly the LRU algorithm. And the third, the number of page faults when we chose the sequence with 32 pages with 4 page frame in memory was 15 for optimal, 16 for FIFO and 17 for LRU. This answer probably depends in the chosen sequence, and it can be a chance in the best event for FIFO.

6. Future Work

LRU resulted to be the best algorithm for page replacement to implement, but it has some disadvantages. In the used algorithm, LRU maintains a linked list of all pages in the memory, in which, the most recently used page is placed at the front, and the least recently used page is placed at the rear. This list must be updated on every memory reference. This includes finding a page in the list, deleting it, and then moving it to the front. It is a very time consuming operation. In future work, we will search for another way to implement LRU. The performance of LRU will be tested with different data structures of algorithms to define which the best is. Then the LRU cache will be implemented.

References

Brehob M. Wagner S. Torng E. Enbody R. (2004). Optimal replacement Is NP-hard for nonstandard caches. IEEE Transactions on Computers. Vol. 53. No. 1. pp. 73-76

Cormen T. H. Leiserson C. E. Rivest R. L. Stein C. (2009). Introduction to algorithms. third edition. MIT press

Chavan A. Nayak K. R. Vora K. D. Purohit M. D. Chawan P. M. (2011). A comparison of page replacement algorithm. IACSIT International Journal of Engineering and Technology. Vol. 3. No. 2

Kavar C. C. Parmar S. S. (2013). Performance analysis of LRU page replacement algorithm. International Journal of Engineering Research and Applications (IJERA) Vol. 3. Issue 1. pp.2070-2076

Klues K. Rhoden B. Zhu Y. Waterman A. Brewer E. (2010). Processes and resource management in a scalable many-core OS. HotPar10. Berkeley. CA

Lo C.T. Srisa-An W. Chang J.M. (2001). A study of page replacement performance in garbage collection heap. Journal of Systems and Software. issue 3. Vol. 58. pp. 235 - 245

Paajanen H. (2007). Page replacement in operating system memory management. Master's Thesis in Information Technology. University of Jyväskylä

Park Seon-yeong Jung D. Kang Jeong-uk Kim Jin-soo Lee J. (2006). Cflru: a replacement algorithm for flash memory. Proceedings of the 2006 international conference on Compilers. architecture

Tanenbaum A. S. (2008). Modern Operating Systems. Pearson International Edition. Chapter 3

Soares L. Stuum M. (2010). FlexSC: Flexible system call scheduling with exception-less system calls. Proceedings of the 9th USENIX conference on Operating systems design and implementation

Windows Internals book. fifth edition. *Memory management* (chapter 9)