International Journal of Information Technology and Management

Vol. VIII, Issue No. XI, February-2015, ISSN 2249-4510

PAGE REPLACEMENT USING FIFO AND LRU

International Journal of Information Technology and Management
Vol. VIII, Issue No. XI, February-2015, ISSN 2249-4510

Page Replacement Using FIFO and LRU

Rekha[1] Mr. Khaushal[2] Dr. Soniya[3]

Abstract – In this paper, a hybrid page replacement algorithm has been represented that has a better performance in average than quondam methods. A major attempt in such an algorithm is to substitute characteristics of some quondam methods engrossed by a new idea. The key concept of proposed scheme is to combine the two algorithms i.e. FIFO and LRU In brief this paper remonstrate an advanced version of Least recently used algorithm and FIFO , which is referred as Hybrid LRU and FIFO.

Keywords: FIFO, LRU Algorithm, Page Replacement Algorithm, Paging

---------------------------♦-----------------------------

INTRODUCTION

The most important part of the operating system is memory management. Main memory is divided into fixed size units called page frames. Each victimized page can be either in secondary memory or in main memory as page frames. The virtual address space [3] is divided into fix size blocks called pages and this division is done by operating system. A CPU generated address is called logical address or virtual address, whereas memory management unit generated address is known as physical address. Before using this logical address, it must be translated to its corresponding physical address. This address translation has been done corresponding to every memory reference, so it is important that it must be fast. A special hardware unit referred to as **Memory Management Unit (MMU)**, is used for such translation. MMU uses address mapping information which is usually located in page tables, to make the translation. If the given virtual address is not mapped to main memory, operating system is trapped by the MMU. This trap is called as **page fault** which gives an opportunity to the operating system to bring the desired page from secondary memory to main memory, and then update the page table correspondingly. In simple words we can say that- **When the processor need to execute a particular page and main memory does not contain that page, this situation is known as PAGE FAULT.** As each and every process has its own virtual address space, the operating system must keep track of all pages and the location of each page used by each process. Whenever any page in main memory is referenced or written to, it is marked accordingly. When a page fault occurs (known as cache miss), the operating system eliminates some page to secondary memory to make space for the incoming page. Whenever a cache miss occurs, the operating system applies page replacement algorithm to choose a page from cache for replacement or eviction to make place for the referenced page. When the selected page is modified while it is in cache (also called as dirty), it must be again written to the RAM also known as write back. If any page will not modified, no write back will be need to the RAM, which results less overhead. A **paging algorithm or page replacement algorithm [4]** is needed to manage paging. A paging algorithm evicts pages from and to the page table when it becomes full. Many algorithms have been developed for such a swapping in which the best-case scenario is to be replaced the page that is not going to be used for the longest time. Since it is difficult to predict the future reference, a better way to choose the correct algorithm is by looking at characteristics of the processes. It is necessary to influence which page should be replace. Evicting a page that may be required in close future can degrade the system performance; because it imposes a reloading time overhead.

PREVIOUS WORK

Page Replacement Algorithm

In a computer operating system paging is used for virtual memory management and **page replacement algorithms** are used to decide which memory pages to page out (swap out, write to disk) when a page of memory requires to be allocated. Paging encounters when a page fault occurs and a free page cannot be used to fulfill that allocation, either because there are no free pages, or because the number of free pages is less than some brink. When any page that was hand-picked for eviction from memory and paged out, is referenced again it has to be rewrite in memory and this includes waiting for I/O culmination. There are a variety of page replacement algorithms. Some of them are described as follows:-

First in, first out (FIFO)

The first-in first-out algorithm [2] is the simplest and oldest algorithm. The idea behind FIFO is to replace a page that is the oldest page in main memory from all the pages. „Replace the page which has been resident from longest period of time." FIFO focuses on the length of a time a page has been in memory rather than how much the page is being used. Figure.1 illustrates an example of the FIFO algorithm. Here it is important to note that since the page table is initially empty; directly three page faults appear to fill the table and after that, a page fault occurs only when any required page is not present in the table at that time and so on.

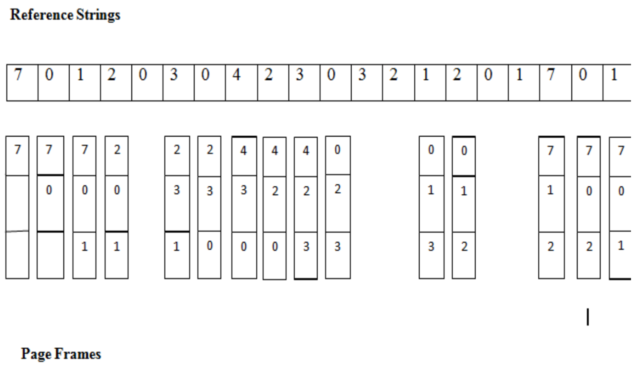

Figure.1: FIFO representation for 3 frames

Least Recently Used (LRU)

The LRU policy is based on the principle of locality which states that program and data references within a process tend to cluster. The LRU page replacement policy chooses that page for replacement which has not been used for the longest time. For a long time, LRU was deliberated to be the most optimum online policy. LRU while being operative is again, not without problems. The first drawback among them is the fact that it is very costly to implement it. In fact, the most costly method in linked with LRU, which facilitates in attaining what it is meant to. This can be a factor for not choosing this algorithm. The second problem with this approach is the difficulty in implementation. LRU policy does nearly as well as an optimal policy, but it is intricate to implement and imposes significant overhead. The LRU is based on the observation that pages that have been used a lot in the last few instructions will probably be utilized a lot again in the next few. Contrarily, pages that have not been utilized for ages will probably remain unused for longest period of time. This idea suggests a realizable algorithm: when a page fault takes place, evict the page that has been unused for the longest period of time. There are a few implementation methods for this algorithm that attempt to diminish the cost yet keep as much of the performance as possible. The most costly method is the linked list method, which utilizes a linked list enclosing all the pages in memory. At the front is the most recently used page and at the back of this list is the least recently used page, which is a very time-consuming process. LRU's weakness is that its performance tends to degenerate under many quite common reference patterns. On the other hand one important advantage of the LRU algorithm is that it is agreeable to full statistical analysis.

Proposed work

We are proposing a methodology in which we are merging two algorithms i.e. FIFO and LRU. By making this hybrid algo we can achieve better page replacement by including advantages of two different algorithms. Output of FIFO will be serve as a input for LRU.

ALGORITHUM:

- Firstly include pages in secondary memory.
- Now shift pages in primary memory as per its usage
- For page replacement from primary memory, replace the page which is sorted as per combination of its entrance and its usage.
- Finally replace that page from primary memory.

FUTURE SCOPE

We can achieve more efficient page replacement by making hybrid of other algorithms. So that we can increase the efficiency of the system and better page replacement .we can also reduce the time of page replacement using by the processor.

CONCLUSION

In this paper we have discussed famous page replacement policies like FIFO, LRU then implemented and compared them to evaluate their efficiency .And combine the two famous algorithms i.e. FIFO and LRU to present the better page replacement.

REFERENCES

[1] O "Neil, J. E., O "Neil, E. P., Weikum, G., "An Optimality Proof of the LRU-K Page Replacement Algorithm", Journal of the ACM, Vol. 46, No. 1, pp. 92- 112, January 1999.

[2] Ali Khosrozadeh, Sanaz Pashmforoush, Abolfazl Akbari, Maryam Bagheri, Neda Beikmahdavi., "Presenting a Novel Page Replacement Algorithm Based on LRU" , Journal of Basic and Applied Scientific Research , 2(10)10377-10383, 2012.

[3] Kim, K., Park, K., "Least Popularity – Per – Byte Replacement Algorithm for a Proxy Cache ", IEEE, pp. 780 – 787, 2001.

[4] S.M. Shamsheer Daula, Dr. K.E Sreenivasa Murthy, G Amjad Khan., “A Throghput Analysis on page replacement algorithm”, International Journal of Engineering Research and Applications (IJERA), ISSN: 2248-9622, Vol. 2, Issue 2, pp.126-13, Mar-Apr 2012.

[5] Jaafar Alghazo, Adil Akaaboune, and Nazeih Botros. Sf-lru cache replacement algorithm. In MTDT, pages 19-24. IEEE Computer Society, 2004

[6] Donghee Lee, Jongmoo Choi, Jong-Hun Kim, Sam H. Noh, Sang Lyul Min, Yookun Cho, and Chong-Sang Kim. Lrfu: A spectrum of policies that subsumes the least recently used and least frequently used policies. IEEE Trans. Computers, 50(12):1352-1361, 2001.

[7] Andrew S. Tanenbaum. Modern Operating Systems. Prentice-Hall, 1992