International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2020 IJSRCSEIT | Volume 6 | Issue 2 | ISSN : 2456-3307
DOI : https://doi.org/10.32628/CSEIT2062123

Hybrid Page Replacement Algorithm in real time Operating System

Pallab Banerjee[1], Biresh Kumar[2], Amarnath Singh[3], Shipra Sinha[4], Medha Sawan[5]

[1,2,3]Assistant Professor, Department of Computer Science and Engineering, Amity University Jharkhand, Ranchi, India

[4,5]B.Tech Scholar, Department of Computer Science and Engineering, Amity University Jharkhand, Ranchi, India

ABSTRACT

Programming codes are of variable length. When the size of codes becomes greater than that of primary memory, the concept of virtual memory comes into play. As the name suggests, virtual memory allows to outstretch the use of primary memory by using storage devices such as disks. The implementation of virtual memory can be done by using the paging approach. Allocation of memory frames to each and every program is done by the operating system while loading them into the memory. Each program is segregated into pages as per the size of frames. Equal size of pages and frames enhance the usability of memory. As, the process or program which is being executed is provided with a certain amount of memory frames; therefore, swap out technique is necessary for the execution of each and every page. This swap out technique is termed as Page Replacement. There are many algorithms proposed to decide which page needs to be replaced from the frames when new pages come. In this paper, we have proposed a new page replacement technique. This new technique is based on the approach of reading and counting of the pages from secondary storage. Whenever the page fault is detected, the needed page is fetched from the secondary storage. This process of accessing the disc is slow as compared to the process in which the required page is retrieved from the primary storage. In the proposed technique, the pages having least occurrence will be replaced by the new page and the pages having same count will be replaced on the basis of LRU page replacement algorithm. In this method, the paged are retrieved from the secondary storage hence, possibility of page hit will be increased and as a result, the execution time of the processes will be decreased as the possibility of page miss will be decreased.

Keywords : Page Replacement, Page Fault, Page Hit, Page Miss, LRU (Least Recently Used)

I. INTRODUCTION

Memory management is the service provided by the Operating System. This manages primary memory. The processes are carried back and forth from secondary storage such as disc to the main memory or primary memory [1]. The process of carrying processes from primary storage to secondary storage for execution to make the memory available for other processes is known as swapping.

A processor can use more memory for execution than the amount of primary memory. This extra amount of memory is called virtual memory. Virtual memory is nothing but a section of hard disk that imitates as primary memory. Virtual memory is generally attained with the demand paging. Segmentation is also used to provide virtual memory. Demand segmentation is one of the techniques to provide virtual memory.

Most commonly used technique paging method is memory management paging in which the memory is parted into fixed size pages [2]. Data can be rapidly accessed through. Operating system duplicates the pages on the main memory from hard disk; hence, whenever a program requires a page, it could be found in the primary memory. Virtual memory uses page table system in a computer's operating system to find the mapping among virtual addresses &physical addresses. Hardware and RAM subsystem uses physical addresses whereas virtual addresses are used to access processes [4].

Whenever a running program tries to reference a page that is not available in RAM, then the processor takes it as an invalid memory reference, or as a page fault and then it relocates control from the program to the OS [22]. Page replacement techniques are the methods by which an Operating System decides which memory pages to be swapped out & write to disk, whenever a page fault is detected by the program. Paging will arise when a page fault occurs and a free page is not to be used for allotment purpose and calculating to reason that pages are not available or the no. of freed pages are lesser than required pages [10].

A page replacement algorithm hits on the less knowledge about obtaining the pages given by the hardware, and then it tries to elect which pages must be replaced to minimize the total number of page misses, during adjusting it with the costs of primary memory & processor time of the algorithm self-[24]. There are various page replacement algorithms which are calculated by executing them on an appropriate string and then calculating the number of page faults. The one with least number of page fault is considered as best algorithm for that particular string.

II. LITERATURE SURVEY

The first-in, first-out (FIFO) algorithm is the simplest page replacement algorithm. In this algorithm, operating system keeps record of each page in memory in the form of a queue, the oldest page is kept in front of the queue.

Whenever a page is to be replaced, the page at the front of the queue, (the oldest page) is selected for the removal. The result of FIFO in practical applications is not satisfactory.

Least recently used (LRU) page replacement is the algorithm in which the page which has not been used for the longest period of time is swapped. We can think of this strategy as the optimal page-replacement algorithm looking back ward in time, rather than forward [26]. The LRU method is regularly used as a page replacement algorithm and is considered to be good but the foremost limitation in putting LRU in operation is that it may require significant hardware support. The difficulty is to decide an order for the frames distinct by the time of last use.

Optimal page replacement algorithm removes the pages which are not to be used for the longest time in future. This algorithm is not easy to implement as it needs future information about the actions of the program which is in execution. It is likely possible to execute optimal page replacement.

Not Frequently Used (NFU) page replacement algorithm, this algorithm also requires future information of actions which is going to be performed by the program in execution. In this method, counter is maintained which stores the information about the repetition of page in future. The page with least count is replaced when there is a need of page replacement.

III. EXISTING ALGORITHM

Existing algorithm is as follows,
The idea is simple, for every reference we do following:

1. If referred page is already present, increment hit count.
2. If not present, find if a page that is never referenced in future. If such a page exists, replace this page with new page. If no such page exists, find a page that is referenced farthest in future. Replace this page with new page.[27]

Consider the following reference string of pages

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	7	0	1

Assume that the frame size is four (F0, F1, F2 and F3). The allocation of frames for the pages in existing methodology is shown below-

HM	0	0	0	0	1	0	1	0	1	1	1	1	1	0	1	1	0	1	1
F0	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	7	7	7
F1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F2			1	1	1	1	1	4	4	4	4	4	4	1	1	1	1	1	1
F3				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Fig. 1: Frame Allocation of Pages in Existing Methodology

HM shown in the above figure denotes hit/miss counts. A one in HM represents a hit while a zero indicates a miss.
The same analysis can be seen in figure 2.

HMC	Existing
Total Pages	19
No. of Hits	11
No. of Missed	8
Hit Ratio	57.89

Fig. 2: Hit/Miss Analysis Using Existing Methodology

IV. SHORTCOMINGS OF EXISTING ALGORITHM

The existing methodology was based on removing the pages which are not used since longest time. As its name imply an optimal page replacement technique is optimal in terms of a smaller number of page faults, which leads to high hit ratio. Along with high hit ratio, it is also known that optimal page replacement technique is not practical because we are not aware of page reference string in advance and also it is difficult to detect error in this algorithm. It sometimes takes a lot of time to remove the least recently used page.

V. PROPOSED METHOD

In this research, we proposed a new concept for page replacement, which is based on reading and counting of pages from the secondary storage. Since, accessing the disk is a time-consuming process and leads to slow processing of data, it is preferable to read a block of data whenever there is a frequent accessing of secondary storage. In this algorithm, a counter will be provided to each page which will count the number of times a single page will going to be repeated and whenever the page fault will occur, instead of reading only the missed frame, the pages equal to the number of frames will be read and the page will be replaced with the pages having least counter value. If the pages will have s.

As we know that disc, access is time consuming because of the complex mechanism of secondary storage, which lead to slow processing of data. It is always better to read a block of data whenever there is frequent disc access. In my research, whenever there will be a page fault, instead of reading a missed page only, I retrieve asset of pages equal to number of frames allotted for that process. By this way, we can definitely minimize number of page miss, which will improve hit ratio too.

VI. PROPOSED ALGORITHM

Our proposed algorithm is given below

Assume that size of reference string is N and allotted number of memory frames are MF

Step 1: Enter length of reference string N and allotted number of memory frames MF.
Step 2: Enter reference string of pages.
Step 3: We will count the number of occurrences of each page and store it in a 2-d array.
Step 4: For first to last position of reference string do steps from 4 to 8.
Step 4: If the marked page is not available in memory frame and memory frame is empty then do step 5, else
if the marked page is not available in memory frames then do step 6, otherwise do step 7.
Step 5: Mark it as page miss and then read a block of next MF pages from the disc as a block retrieval and fill all the allotted frames at once and go to step 8.
Step 6: Mark it as page miss and then read MF pages from the disc as a block retrieval and replace that page with the page with minimum count present in MF and go to step 8.
Step 7: Just mark it as page hit.
Step 8: Reduce the count of string at current position by 1.

Consider the following reference string of pages-

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	7	0	1

Let consider the frame size is four (F0, F1, F2 and F3). The allocation of frames for the pages in proposed methodology is show below-

Page	0	1	2	3	4	7
Count	6	3	4	3	1	2

A 2-d array containing the number the number of occurrences of each page in the string.

HM	0	1	1	1	1	0	1	0	1	1	1	1	1	0	1	1	0	1	1
F0	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	7	7	7
F1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F2	1	1	1	1	1	1	1	4	4	4	4	4	4	1	1	1	1	1	1
F3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Fig. 3: Working of Proposed Methodology

In the figure 3, we can see, first reference string is 7 which is not found in allotted frames. It is a page miss, thus as per the proposed algorithm, we have to read next four distinct pages (7, 0, 1, 2). After filling all four frames, there will be page hit for the next page references (0, 1, 2, and 0). There will be again a page fault for a page reference 3. Then we will find the page in MF with minimum count of occurrence, here it is 7, and replace its place with 3 and reduce its count.

And count array will look as follow:

Page	0	1	2	3	4	7
Count	4	2	3	2	1	1

Similarly, we will perform further.

HM shown in above figure denotes hit/miss counts. A one in HM represents a hit while a zero indicates a miss.

As per the output,
The total page references = 19 Total number of hits = 14
Total Number of miss= 5
Hit Ratio = [Total Hits / (Total Hit + Miss)] x 100 So hit ratio= [14/19] x 100 = 73.68 %

VII. RESULT AND ANALYSIS

The results are analyzed for four memory frames. The number of hits using existing methodology for

predefined reference string is 11, but using proposed method is 14. The hit ratio using existing method for default reference string is 57.89%, but using proposed it increases to 73.68 %, which is a big difference. The proposed methodology will depict better result when number of memory frames are increased.

The result analysis of existing Vs proposed algorithm is shown using following snapshot

HMC	Existing	Proposed
Total Pages	19	19
No. of Hits	11	14
No. of Missed	8	5
Hit Ratio	57.89	73.68

Fig. 4 : Result Analysis Using Existing Vs Proposed Methodology

The same analysis is shown using bar chart-

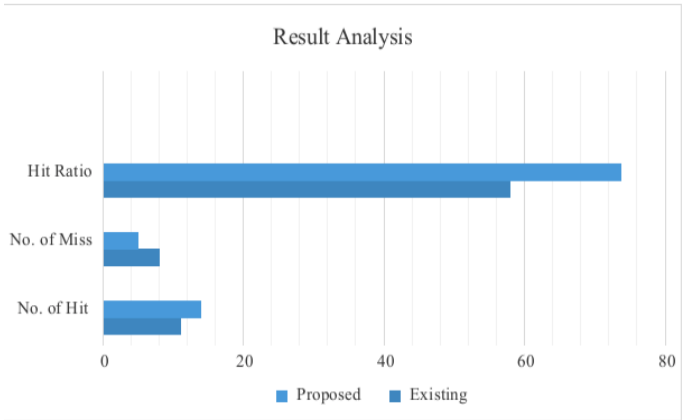

Fig. 5 : Result Analysis of Existing Vs Proposed Methodology Using Bar chart

VIII. CONCLUSION

In this research, a new concept of page replacement has been introduced, which is based on reading and counting pages from secondary storage. This reading and counting is obvious when there is frequent disc access. With the help of proposed methodology, we can find maximum pages in memory frames, which result in high hit ratio. When the existing approach is compared to the proposed approach it is clear that proposed algorithm provides better results. If the frame size will be increased then also proposed algorithm will provide better results. Although the proposed algorithm shows better, result but there is always a need of improvement. In future, the same methodology can be improved by applying some concept, which will reduce the number of page replacement.

IX. REFERENCES

[1]. Sanjay Kumar Panda and Saurav Kumar Bhoi, "An Effective Round Robin Algorithm using Min-Max Dispersion Measure", International Journal on Computer Science and Engineering, 4(1), pp. 45-53, January2012.

[2]. Sorav Bansal and Dharmendra S. Modha CAR: Clockwith Adap- tive Replacement FAST'04 - 3rd USENIX Conference on File and Storage Technologies,2004.

[3]. G. Glass and P. Cao, Adaptive Page Replacement Basedon Memory Reference Behavior, Proceedings of 1997ACM SIG- METRICS Conference, May 1997, pp.115-126.

[4]. S. Jiang, and X. Zhang, "LIRS: An Efficient Policy to improve Buffer Cache Performance", IEEE Transactions on Computers, pp. 939-952,2005

[5]. Nimrod Megiddo and Dharmendra S. Modha ARC: ASelf-tuning, Low Overhead Replacement Cache USENIX File and Storage Technologies Conference (FAST), San Francisco, CA,2003.

[6]. N. Meigiddo, and D. S. Modha, "ARC: A Self-Tuning, Low overhead Replacement Cache", IEEE Transactions on Computers, pp. 58-65,2004.

[7]. S. Bansal, and D. Modha, "CAR: Clock with Adaptive Re- placement", FAST-'04 Proceedings

of the 3rd USENIX Confer- ence on File and Storage Technologies, pp.187-200,2004.

[8]. Theodore Johnson and Dennis Shasha. 2q: a lowover head high performance buffer management replacement algorithm In Proceedings of the Twentieth International Conference on. very Large Databases, pp.439-450, Santiago, Chile,1994.

[9]. J. E. O'neil, P. E. O'neil and G. Weikum, "An optimality Proof of the LRU-K Page Replacement Algorithm", Journal of the ACM, pp. 92-112,1999.

[10]. A. Janapsatya, A. Ignjatovic, J. Peddersen and S. Parameswaran, "Dueling CLOCK: Adaptive cache replacement policy based on the CLOCK algorithm", Design, Automation and Test in Europe Conference and Exhibition, pp. 920-925,2010

[11]. Implementation of a page replacement algorithm with temporal filtering for Linux, vashundra rathod, pramia chavan, journal of engineering &applied sciences volume 2, no. 6, june2013.

[12]. S. Jiang, X. Zhang, and F. Chen, "CLOCK-Pro: An Effective Im- provement of the CLOCK Replacement", ATEC '05 Proceedings of the annual conference on USENIX Anuual Technical Confer- ence, pp. 35,2005.

[13]. Kaveh Samiee," WRP: Weighting Replacement Policy to Im- prove Cache Performance", International Journal of Hybrid In- formation Technology, Vol.2, No.2, April,2009.

[14]. Song Jiang, Feng Chen and Xiaodong Zhang, CLOCK Pro: An Effective Improvement of the CLOCK Replacement, USENIX Annual Technical Conference,2005.

[15]. Pooja khulbe, Shruti pant, "HYBRID LRU Page Replacement Algorithm" , International Journal of Computer Applications (0975 – 8887) Volume 91 – No.16, April2014.

[16]. Performance analysis of LRU page replacement algorithm. Inter- national Journal of Engineering Research and Applications (IJE-RA) Vol. 3. Issue 1. pp.2070-2076 Klues K. Rhoden B. Zhu Y. Waterman A. Brewer E. (2010).

[17]. A. S. Sumant, and P. M. Chawan, "Virtual Memory Management Techniques in 2.6 Linux kernel and challenges", IASCIT Interna- tional Journal of Engineering and Technology, pp. 157-160,2010.

[18]. A. Janapsatya, A. Ignjatovic, J. Peddersen and S.Parameswaran, "Dueling CLOCK: daptive cache replacement policy based on the CLOCK algorithm", Design, Automation and Test in Europe Conference and Exhibition,pp. 920-925,2010.

[19]. Amit S. Chavan, Kartik R. Nayak, Keval D. Vora, Manish D.Purohit and Pramila M. Chawan, " A Comparison of PageRe- placement Algorithms", IACSIT International Journal ofEngi- neering and Technology, Vol.3, No.2, April 2011pp.171-174.

[20]. Jisha.P. Abraham, Sheena Mathew "A novel approach to im- prove processor performance with page replacement techniques" Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014,3-5 December2014.

[21]. Song Jiang and Xiaodong Zhang, Token-ordered LRU: an effec- tive page replacement policy and its implementation in Linux sys- tems, Performance Evaluation 60 5–29,2005.

[22]. Mr.C.C.Kavar, Mr. S.S.Parmar "Performance Analysis of LRU Page Replacement Algorithm with Reference to different Data Structure"International Journal of Engineering Research and Applications (IJERA) Vol. 3, Issue 1, January – February2013,pp.2070-2076.

[23]. A comparison of page replacement algorithm. IACSIT Interna- tional Journal of Engineering and Technology. Vol. 3. No. 2 Kavar C. C. Parmar S. S. (2013).

[24]. Abraham Silberschatz, Peter B. Galvin and Greg Gagne, Operat- ing System Concepts (UK: Wiley,2010).

[25]. Page Replacement, S. Jananee, ISSN 2348-1196 (print) Interna- tional Journal of Computer Science and Information Technology Research ISSN 2348-120X (online) Vol. 2, Issue 3, pp: (90-99), Month: July - September2014.

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

[27]. Optimal Page Replacement Algorithm,https://www.geeksforgeeks.org/optimal-page-replacement-algorithm

Cite this article as :

Pallab Banerjee, Biresh Kumar, Amarnath Singh, Shipra Sinha, Medha Sawan, "Hybrid Page Replacement Algorithm in real time Operating System", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 6 Issue 2, pp. 433-439, March-April 2020. Available at doi : https://doi.org/10.32628/CSEIT2062123
Journal URL : http://ijsrcseit.com/CSEIT2062123