ISSN:0975-9646

Ruchi Gairola et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 8 (4) , 2017, 479-482

Design & Implementation of a Page Replacement Algorithm Using Block Reading

Ruchi Gairola
M.Tech Scholar, UTU, Dehradun
Uttarakhand Technical University, Dehradun

Sanjay Kumar
Assistant Professor - CSE
Uttarakhand Technical University, Dehradun

***Abstract*— The software developers write programming codes of any length without concerning of primary memory available with the users. It is possible by using the concept of virtual memory. As the name implies, virtual memory is a concept of executing a programming code of any size even having a primary memory of lesser size than the size of program to be executed. The virtual memory is implemented with the help of concept called paging. The operating system allocates a number of memory frames for each and every program to be executed. The programming code is equally divided into a number of pages. The size of pages and memory frames are kept equal for the better utilization of the memory. As every process is allotted a limited number of memory frames, the need of page replacement is obvious. To overcome this limitation, a number of page replacement techniques are proposed by the researchers. In this thesis I have proposed an improved page replacement technique which is based on the concept of block reading from the secondary storage. The disc access is very slow as compared to primary memory access. Whenever there is a page fault, the required page is accessed from the secondary storage. The frequent page faults increase the execution time of process. As per the proposed methodology, a number of pages equal to the allotted memory frames are read every time when there is a page fault. After reading a block of pages, it definitely increases the possibilities of page hit and as a result it will improve the hit ratio for the processes.**

***Keywords*- Page replacement, Page fault, Page hit, Page miss, Hit ratio, Block reading.**

I. INTRODUCTION

Operating system provides a service known as memory management, which governs and guide primary memory also manages and handles main memory, it moves processes between main memory and disc during execution by back forth [1]. The process in which we temporarily moves process from primary memory to the hard disk or secondary memory so the memory be available for other processes, the process is known as swapping.

A computer can locate extra memory, than the amount of manually equipped on the system. This extraneous memory is literally called virtual memory & it is indeed a section of a hard disc that is set up to imitate the computer's RAM. Virtual memory is generally achieved with the demand paging. It may also be carried out in a segmentation system. For providing virtual memory, Demand segmentation is to be used.

A memory management method paging is frequently used in which the memory is parted into fixed size pages[8]. Paging is used for accessing data rapidly. Whenever a program requires a page, it could be found in the primary memory as if the Operating System duplicates a certain no. of pages on the main memory from hard disk. It grants the physical address space of a process to be non-contiguous. A page table is the data structure, which is used by a virtual memory system in a computer operating system to fund the mapping within the virtual addresses and physical addresses. Virtual addresses are used by the accessing process, while physical addresses usedup by the hardware & most categorically, by the RAM sub-system[9]. Whenever a program attempt 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 relocate control from the program to the OS[11].

Page replacement techniques are the methods by using which an Operating System concludes which memory pages to be swapped out and write to disk, whenever a page of main memory is required to be allocated. Paging will arise when a page fault occurs &a free page is not to be used for allotment purpose, & calculating to reason that pages are not available or the no. of freed pages is lesser than required pages [14].

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-[15]. We have many different page replacement algorithms. We calculate an algorithm by executing it on a appropriate string of memory reference and checking the number of page faults.

II. LITERATURE SURVEY

The first-in, first-out (FIFO) page replacement algorithm is a less-overhead algorithm that entails little bookkeeping on the part of the operating system. As we know by the name - the operating system set track of each page in memory in the form of a queue, with the one comes late placed at last, and the one comes first will placed in front. The operating system assists a list of all pages presently in memory, with that page which is at the head of the list the oldest one and the page at the tail the most topical arrival. Whenever a page is to be swapped out, the page at the front of the queue (the oldest page) is considered. While FIFO is cheap and instinctive, it results poorly in practical application.

Least recently used (LRU) page replacement, This algorithm replaces the page that has not been used for the longest period of time. We can think of this strategy as the optimal page-replacement algorithm looking back ward in time, rather than forward [20]. The LRU policy is regularly used as a page replacement algorithm and is well thought-out to be good. The foremost problem is how to put into operation LRU replacement. An LRU page-replacement algorithm may involve significant hardware support. The difficulty is to decide an order for the frames distinct by the time of last use.

Optimal page replacement,[21] The optimal page algorithm merely removes, the page with utmost no. of such information implying that it will be required in the most isolated future. This

algorithm was launch long back & is not easy to implement for the reason that it requires future information of the program actions. It is likely possible to execute optimal page replacement.
Not Recently Used (NRU) page replacement algorithm[22],in this algorithm it is important that it requires that each page must contain 2 extra status bits 'R' and 'M' called reference bit & change bit respectively. The reference bit(R) is repeatedly set to 1 at whatever time the page is requested. The change bit (M) is set to 1 every time the page is customized, These bits are stored in the PMT and are reorganized on each & every memory reference. Whenever a page fault occurs the memory manager inspects all the pages & divides them into 4 classes based on R and M bits.

III. EXISTING ALGORITHM

Existing algorithm is as follows, first the author determine number of pages. Let us say this is denoted by the value 'n'. Now we create 'n' count variables, say c1, c2, and c3 upto cn. Now we take the reference string, count each value, and add it to the count of that corresponding value [30]. For example if the reference string value is 1,1,3,2,0,5,6,2,4 then author have 6 count variables and their values are,c1=2,c2=2.c3=1,c4=1,c5=1,c6=1 and c0=1.Now suppose we have 4 frames, so first 1 is entered to the frame and c1 is now equal to 1. Next 1 is already there, hence only the value of c1 changes and is equal to 0. Next 3 is requested now 3 added to the frame, now count for c3 is 0. Similarly, 2 and 0 are added. When 5 to be enter, page fault occurs. For placing it the value with minimum no. of count is to be replaced. If an ambiguous case occurs then the LRU algorithm or FIFO can be followed to remove a page.

Consider the following reference string of pages-

1	1	3	2	0	5	6	2	4	5

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	1	0	0	0	0	0	1	0	1
F0	1	1	1	1	1	5	5	5	5	5
F1			3	3	3	3	6	6	4	4
F2				2	2	2	2	2	2	2
F3					0	0	0	0	0	0

Figure 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	10
No.of Hits	3
No.of Missed	7
Hit Ratio	30.0000...

Figure 2 Hit/Miss Analysis Using Existing Methodology

IV. SHORTCOMING OF EXISTING ALGORITHM

The existing methodology was based on count based page replacement technique, which was similar to the optimal page replacement. As its name imply an optimal page replacement technique is optimal in terms of less number of page faults, which lead 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.

V. PROPOSED METHOD

In this research, we proposed a new concept for page replacement, which is based on block reading of pages from the secondary storage. As we know that disc, access is time consuming because of the complex mechanism of secondary storage, which leads 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: for first to last position of reference string do steps from 4 to 6.
Step 4: if the marked page is not available in memory frames then do step 5, otherwise do step 6.
Step 5: mark it as page miss and then read a block of next MF pages from the disc as block retrieval and fill all the allotted frames at once.
Step 6: just mark it as page hit.

Consider the following reference string of pages-

1	1	3	2	0	5	6	2	4	5

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

HM	0	1	1	1	1	0	1	1	1	1
F0	1	1	1	1	1	5	5	5	5	5
F1	3	3	3	3	3	6	6	6	6	6
F2	2	2	2	2	2	2	2	2	2	2
F3	0	0	0	0	0	4	4	4	4	4

Figure 3 Working of Proposed Methodology

In the figure 3, we can see, first reference string is 1 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 (1, 3, 2, 0). After filling up the all four frames, there will be page hit for the next page references (1, 3, 2, and 0). There will be again a page fault for a page reference 5. Then with similar method, next four pages will be read (5, 6, 2 and 4), after that all coming pages will be found resulting as page hit.

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

As per the output,
The total page references =10
Total number of hits =8
Total Number of miss=2
Hit Ratio= [Total Hits/(Total Hit + Miss)] x 100
So hit ratio= [8/10] x 100 = 80 %

VII. RESULTS AND ANALYSIS

The results are analysed for four memory frames. The number of hits using existing methodology for predefined reference string is 3, but using proposed method it is increases to 8, which is a significant change. The hit ratio using existing method for predefined reference string is 30 .00 %, but using proposed it increases to 80.0 %, which is a big difference. The proposed methodology will depict better result when number of memory frames increase.

The result analysis of existing Vs proposed algorithm is shown in figure 4.

HMC	Existing	Proposed
Total Pages	10	10
No.of Hits	3	8
No.of Missed	7	2
Hit Ratio	30.0000...	80.0

Figure 4 Result Analysis Using Existing Vs Proposed Methodology

The same analysis is shown using bar chart-

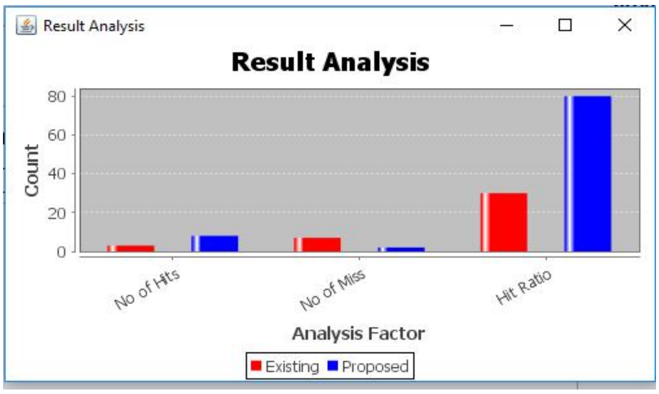

Figure 5 Result Analysis of Existing Vs Proposed Methodology Using Bar chart

VIII. CONCLUSION

In this research, we have preserved a new concept for page replacement, which is based on block reading from secondary storage. The concept of block reading is obvious when there is frequent disc access. With the help of proposed methodology, we can found maximum pages in memory frames, which result high hit ratio. If we compare the proposed methodology with existing one, we found that the proposed methodology provide better results. As usual, the proposed method will provide better result when we allot more number of memory frames. 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. The proposed algorithm can be improved by providing a hybrid mechanism, which uses existing algorithms (like first in first out, least recently used, optimal page replacement, etc.) also.

REFERENCES

[1] G. Glass and P. Cao, Adaptive Page Replacement Basedon Memory Reference Behavior, Proceedings of 1997ACM SIG- METRICS Conference, May 1997, pp. 115-126.
[2] 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.
[3] Nimrod Megiddo and Dharmendra S. Modha ARC: ASelf-tuning, Low Overhead Replacement Cache USENIX File and Storage Technologies Conference(FAST), San Francisco, CA, 2003.
[4] N. Meigiddo, and D. S. Modha, "ARC: A Self-Tuning, Low overhead Replacement Cache", IEEE Transactions on Computers, pp. 58-65, 2004.
[5] S. Bansal, and D. Modha, "CAR: Clock with Adaptive Replacement", FAST-'04 Proceedings of the 3rd USENIX Conference on File and Storage Technologies, pp. 187-200,2004.
[6] Sorav Bansal and Dharmendra S. Modha CAR: Clockwith Adaptive Replacement FAST'04 - 3rd USENIX Conference on File and Storage Technologies, 2004.
[7] S. Jiang, and X. Zhang, "LIRS: An Efficient Policy to improve Buffer Cache Performance", IEEE Transactions on Computers, pp. 939-952, 2005.
[8] Song Jiang, Feng Chen and Xiaodong Zhang, CLOCK Pro: An Effective Improvement of the CLOCK Replacement, USENIX Annual Technical Conference,2005.
[9] Song Jiang and Xiaodong Zhang, Token-ordered LRU: an effective page replacement policy and its implementation in Linux systems, Performance Evaluation 60 5–29, 2005.
[10] S. Jiang, X. Zhang, and F. Chen, "CLOCK-Pro: An Effective Improvement of the CLOCK Replacement", ATEC '05 Proceedings of the annual conference on USENIX Anuual Technical Conference, pp. 35, 2005.
[11] Kaveh Samiee, "WRP: Weighting Replacement Policy to Improve Cache Performance", International Journal of Hybrid Information Technology,Vol.2,No.2, April, 2009.
[12] 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.
[13] Abraham Silberschatz, Peter B. Galvin and Greg Gagne, Operating System Concepts (UK: Wiley, 2010).
[14] 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).
[15] A. S. Sumant, and P. M. Chawan, "Virtual Memory Management Techniques in 2.6 Linux kernel and challenges", IASCIT International Journal of Engineering and Technology, pp. 157-160, 2010.

[16] 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.

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

[18] 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.

[19] Implementation of a page replacement algorithm with temporal filtering for Linux, vashundra rathod, pramia chavan, journal of engineering & applied sciences volume 2, no. 6, june 2013.

[20] 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.

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

[22] Pooja khulbe, Shruti pant, “HYBRID LRU Page Replacement Algorithm” , International Journal of Computer Applications (0975 – 8887) Volume 91 – No.16, April 2014.

[23] Page Replacement, S. Jananee, ISSN 2348-1196 (print) International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online) Vol. 2, Issue 3, pp: (90-99), Month: July - September 2014.

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

[25] Hasan M H Owda ,Munam Ali Shah, AbuelgasimIbrahimMusa, ManzoorIlahiTamimy " A Comparison of Page Replacement Algorithms in Linux Memory Management" International Journal of Computer and Information Technology (ISSN: 2279 – 0764) Volume 03 – Issue 03, May2014 pp. 565-569,2015.

[26] Anvita Saxena, A Study of Page Replacement Algorithms, International Journal of Engineering Research and General Science, 2(4), 2014, 385-388,2015.

[27] Manisha Koranga and Nisha Koranga, Analysis on Page Replacement Algorithms with Variable Number of Frames, International Journal Of Advanced Research in Computer Science and Software Engineering, 4(7), 2014, 403-411,2015.

[28] Genta Rexha, Erand Elmazi and Igli Tafa, A Comparison of Three Page Replacement Algorithms: FIFO, LRU and Optimal, Academic Journal of Interdisciplinary Studies, 4(2), 2015, 56-62, 2016.

[29] Mahesh Kumar M R and Renuka Rajendra B, AN INPUT ENHANCEMENT TECHNIQUE TO MAXIMIZE THE PERFORMANCE OF PAGE REPLACEMENT ALGORITHMS, International Journal of Research in Engineering and Technology, 4(6), 2015, 302-307,2016.

[30] Shreyank Gowda, “Count based page replacement technique” Proceedings of The IIER International Conference, Los Angeles, USA, 7th April 2016, ISBN: 978-93-85973-57-4, 2016