# SIMULATE AND IMPLEMENTATION OF PAGE REPLACEMENT TECHNIQUES THROUGH MEMORY ADDRESSES

**Brijesh Kumar Mishra**
*(Deptt. Of CSE, Monad University, Pilkhuwa, Hapur, INDIA)*

## ABSTRACT

*There are many page replacement Algorithms such as LRU, FIFO, Optimal, WS, NRU, NFU, Random, Second chance, Clock, CAR, and ARC etc. are available in memory management. In this paper we try to attempts the analysis and comparative knowledge between FIFO, LRU, Optimal, Random and Second chance page replacement algorithms. And also analysis the efficiency of processor when the page miss situation enters in memory through the memory addresses and standard traces. Our experimental results demonstrate the Performance of the many page replacement techniques in providing the less page miss and small cost of overhead.*

***Keywords - Cache Performance, FIFO, Hit ratio, LRU, Memory Management, Random, SC, Virtual Memory.***

## I. INTRODUCTION

The Main memory provides an important and limited resource in a computer system. So by this importance main memory has essential amount of demand in the past decades. It is also known by everyone that processor and main memory has much faster speed compared to secondary memories. If main memory has enough space for other program then it will shared this space for use by other users. When occupied fraction memory from one programs not sufficient for program execution then to remove this problem we use the concept of virtual memory. A virtual memory system uses efficient and less overhead page replacement algorithms to decide which pages swap out from memory when a page miss occurred. The Pages are carried out into main memory only when the processor demands them for execution of process, this is called demand paging. The virtual memory allows the execution of process that does not exist completely inside main memory. This memory plays a vital role in page replacement techniques to swap in swap out of required pages in memory.

## II. PAGE REPLACEMENT

The page replacement policy is used to select the page in memory that will be replaced when a new page brought in. When a user executes a program then page miss situation occurs. The page replacement takes the following approaches for use-

**(I)** If there is no free space in frame.

**(II)** If it finds some free page in memory that is not really in use then swap out.

**(III)**The same type of page brought into memory in several time.

Page replacement techniques clarify that which memory pages will be page out (swap out, write to disk) when memory need to be allocated it. Paging concept arises when a page miss occurs and a there is no free page on the memory to satisfy the allocation. Page replacement algorithms are divided in to two types-

**(1) Local page replacement:-**When a process found that a page miss occurred on same Process for Page replacement is called local page replacement.

**(2) Global page replacement**: - This is free to select any page in memory.

**First-In, First-Out page replacement (FIFO):** It is a simple page-replacement algorithm. The first-in, first-out (FIFO) is a low-overhead algorithm. The operating system maintains all the pages in memory in a queue pattern, with the most recent arrival at the back, and the earliest arrival in front. When a page needs to be replaced, the page at the front of the queue (the oldest page) is selected.

**Least Recently Used (LRU) page replacement:** The least recently used page replacement algorithm, though similar in name to NRU, differs in the fact that LRU keeps track of page usage over a short period of time, while NRU just looks at the usage in the last clock interval. LRU works on the idea that pages that have been most heavily used in the past few instructions are most likely to be used heavily in the next few instructions too. It is rather expensive to implement in practice. It discards page that has not been accessed in longest time. Use (recent) past as a predictor of the future. LRU replacement associates with each page the time of that page's last use. When a page must be replaced, it chooses the page that has not been used for the longest period of time.

**Second Chance (SC) page replacement algorithm:** A modified form of the FIFO page replacement algorithm, known as the second chance page replacement algorithm relatively better than FIFO at little cost for the improvement. It works by looking at the front of the queue as FIFO does, but instead of immediately paging out that page, it checks to see if its referenced bit is set. If it is not set, the page is swapped out. Otherwise, the referenced bit is cleared, the page is inserted at the back of the queue and this process is repeated. This can also be thought of as a circular queue. If all the pages have their referenced bit set, on the second encounter of the first page in the list, that page will be swapped out, as it now has its referenced bit cleared.

**Random page replacement algorithm**: Random replacement algorithm replaces a random page in memory. This eliminates the overhead cost of tracking page references. Usually it fares better than FIFO, and for looping memory references it is better than LRU, although generally LRU performs better in practice. Throw out a random page.

**(1)** Obviously not the best scheme

**(2)** Although very easy to implement

Probably the simplest page replacement algorithm is the replacement of a random page. If a frequently used page is evicted, the performance may suffer. For example, some page, that contains program initialization code which may never be needed again during the program execution, could be evicted instead. So there are Performance benefits available with choosing the right page.

## III. RESULT ANALYSIS:

We draw a page miss table and graph for each replacement techniques, Draw a page hit table and graph for replacement techniques and Draw AHR table and graph for replacement techniques. After that we measure the performance analysis.

| Frame Size | FIFO | LRU | RANDOM | SECOND CHANCE |
|---|---|---|---|---|
| 2 | 72.27 | 73.42 | 70.8 | 73.02 |
| 4 | 77.25 | 78.07 | 76.12 | 78.07 |
| 8 | 80.70 | 81.00 | 78.8 | 81.25 |
| 16 | 83.22 | 83.70 | 81.17 | 83.67 |
| 32 | 85.22 | 85.82 | 83.57 | 85.65 |
| 64 | 86.90 | 87.07 | 85.65 | 86.82 |
| 128 | 88.00 | 88.20 | 87.42 | 88.1 |
| 256 | 89.20 | 89.25 | 89.22 | 89.2 |
| 512 | 89.92 | 89.92 | 89.92 | 89.92 |
| 1024 | 89.92 | 89.92 | 89.92 | 89.92 |
| 2048 | 89.92 | 89.92 | 89.92 | 89.92 |
| Avg. AHR | 84.77 | 85.12 | 83.86 | 85.05 |

Table 1: Table for % Hit Ratio for algorithms using bzip trace

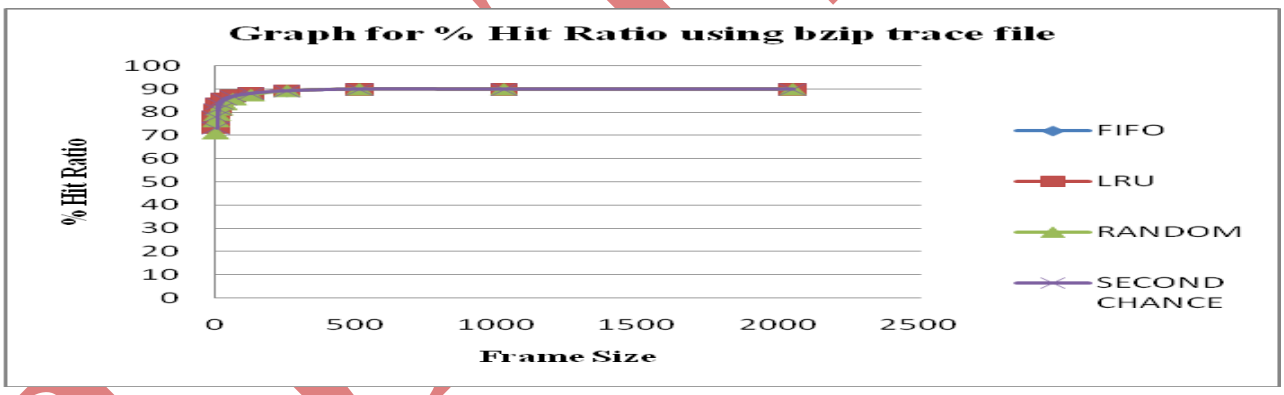


Fig. 1: Graph for % Hit Ratio for algorithms using bzip trace

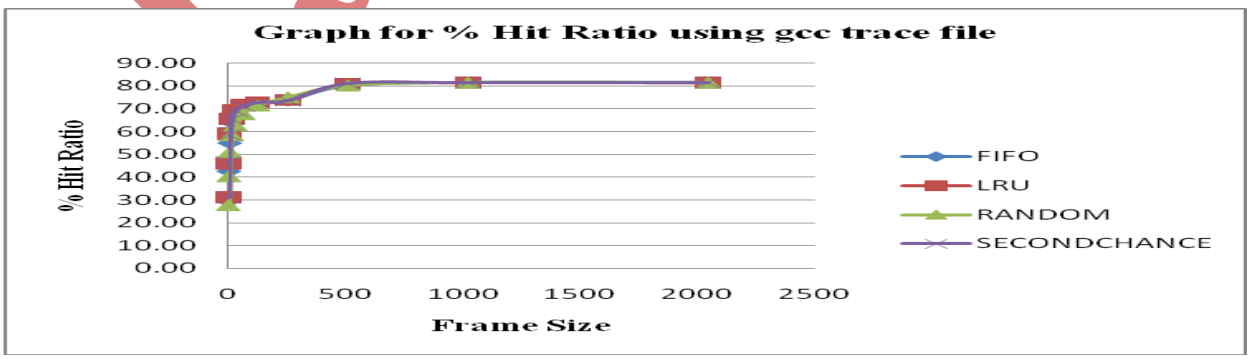


Fig. 2: Graph for % Hit Ratio for algorithms using gcc trace

| Frame Size | FIFO | LRU | RANDOM | SECOND CHANCE |
|---|---|---|---|---|
| 2 | 29.23 | 30.93 | 27.68 | 31.03 |
| 4 | 42.45 | 45.90 | 40.35 | 46.83 |
| 8 | 54.88 | 59.13 | 50.90 | 58.40 |
| 16 | 63.80 | 65.43 | 58.35 | 65.45 |
| 32 | 68.15 | 69.28 | 62.65 | 69.13 |
| 64 | 70.80 | 71.53 | 67.68 | 71.25 |
| 128 | 72.28 | 72.85 | 71.43 | 72.78 |
| 256 | 73.58 | 73.78 | 74.98 | 73.63 |
| 512 | 80.20 | 80.90 | 80.23 | 81.18 |
| 1024 | 81.50 | 81.50 | 81.50 | 81.50 |
| 2048 | 81.50 | 81.50 | 81.50 | 81.50 |
| Avg. AHR | 65.31 | 66.61 | 63.39 | 66.61 |

Table 2: Table for % Hit Ratio for algorithms using gcc trace

| Frame Size | FIFO | LRU | RANDOM | SECONDCHANCE |
|---|---|---|---|---|
| 2 | 40.05 | 44.75 | 38.35 | 43.52 |
| 4 | 51.85 | 58.70 | 50.25 | 58.67 |
| 8 | 61.32 | 64.45 | 61.69 | 65.50 |
| 16 | 72.97 | 78.75 | 77.10 | 80.45 |
| 32 | 93.37 | 96.65 | 93.27 | 96.75 |
| 64 | 98.45 | 98.45 | 98.45 | 98.45 |
| 128 | 98.45 | 98.45 | 98.45 | 98.45 |
| 256 | 98.45 | 98.45 | 98.45 | 98.45 |
| 512 | 98.45 | 98.45 | 98.45 | 98.45 |
| 1024 | 98.45 | 98.45 | 98.45 | 98.45 |
| 2048 | 98.45 | 98.45 | 98.45 | 98.45 |
| Avg. AHR | 82.75 | 84.91 | 82.85 | 85.05 |

Table 3: Table for % Hit Ratio for algorithms using swim trace

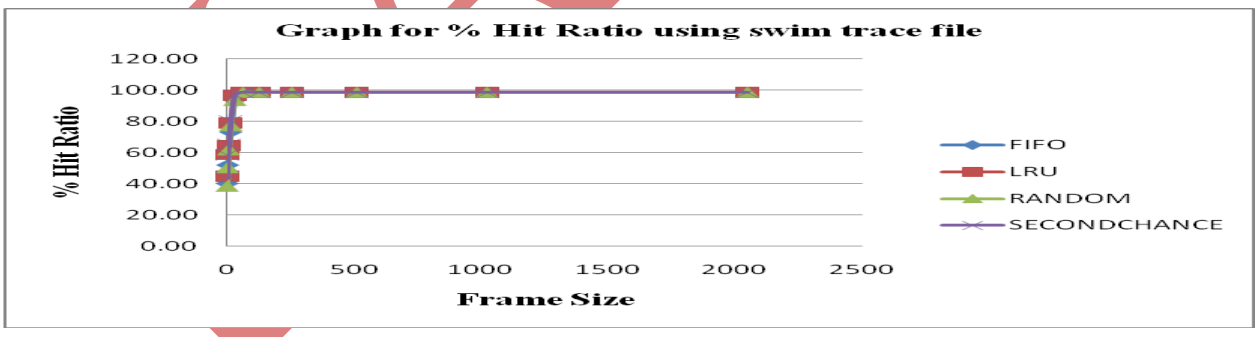


Fig. 3: Graph for % Hit Ratio for algorithms using swim

Now we observed the performance of page replacement algorithms using traces bzip, gcc, and swim. Their conclusions are as follow-

**(1)** When we are using bzip trace for all P.R.A. then we observed that LRU (85.12) performs better than other algorithms. After that Second Chance (85.05), FIFO (84.77) and Random (83.86) respectively are better. Thus LRU P.R.A. is best for page replacement when bzip trace will use.

**(2)** When we are using gcc trace for all P.R.A. then we observed that LRU (66.61) performs better than other algorithms. After that Second Chance (66.61), FIFO (65.31) and Random (63.39) respectively are better. Thus LRU P.R.A. is best for page replacement when gcc trace will use.

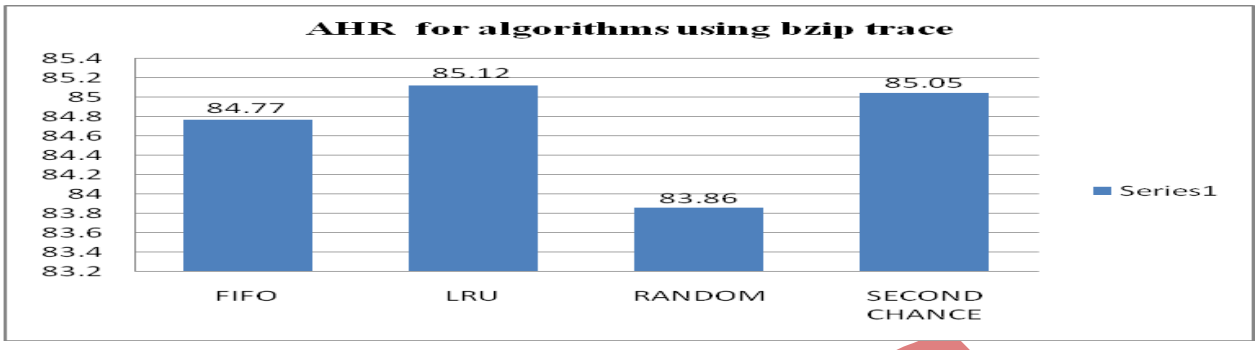


Fig 4: Graph for AHR for algorithms using bzip trace

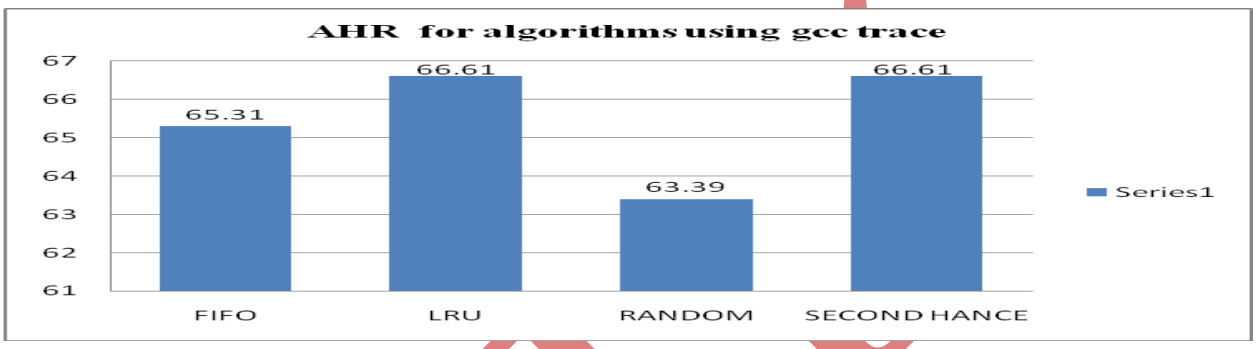


Fig. 5: Graph for % Hit Ratio for algorithms using gcc trace

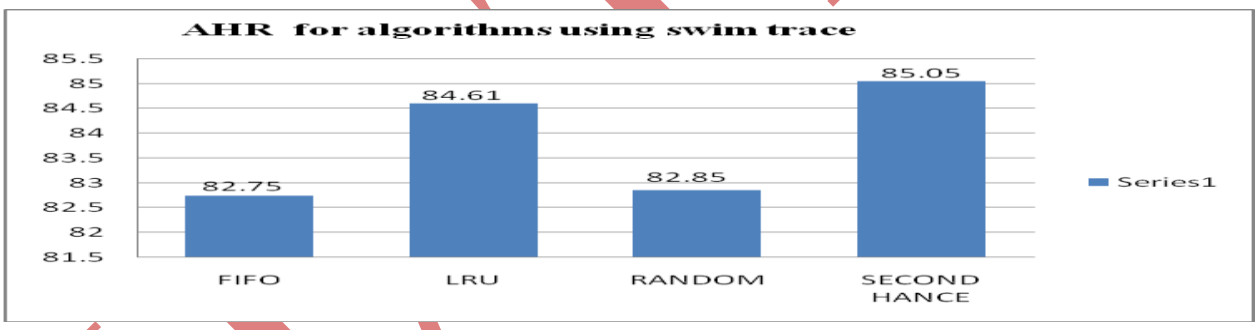


Fig. 6: Graph for AHR for algorithms using swim trace

**(3)** When we are using swim trace for all P.R.A. then there is some difference, we observed that Second Chance (65.05) performs better than other algorithms. After that LRU (84.61), Random (82.85) and FIFO (82.75) respectively are better. Thus Second Chance P.R.A. is best for page replacement when swim trace will use.

## IV. CONCLUSION

Therefore from all conclusions we observed that the first most popular algorithm is used for bzip trace LRU (85.12) and Second Chance (85.05). The second most popular algorithms is used for gcc trace LRU (66.61) and Second Chance (66.61)). These two algorithms provide low overhead on memory and processor. If we are taking swim trace then Second Chance (65.05) and LRU (84.61). But taking overall performance these two algorithms LRU and Second Chance play a vital and effective role in page replacement algorithm. We also observed that if page hit increase then the overhead on memory as well as processor will reduces and performance will be better as previous.

## V. FUTURE WORK:-

From above results and performance analysis we observed that which algorithm is based for page replacement and will reduce the low overhead on memory as well as processor. In future vision we make an algorithm using traces which make faster page replacement and will compare the performance and overhead with exiting algorithms FIFO, LRU, Second Chance and Random mention in this thesis.

## REFERENCES:

**[1]** Amit S. Chavan, Kartik R. Nayak, Keval D. Vora, Manish D. Purohit and Pramila M. Chawan," A Comparison of Page Replacement Algorithms" IACSIT International Journal of Engineering and Technology, Vol.3, No.2, April 2011.

**[2]**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) ISSN: 2248-9622 www.ijera.com Vol. 3, Issue 1, January -February 2013, pp.2070-2076

**[3]** Debabrata Swain, Bancha Nidhi Dash, Debendra O Shamkuwar, Debabala Swain," Analysis and Predictability of Page Replacement Techniques towards Optimized Performance", IRCTITCS, 2011, pp. 12-16.

**[4]** S.M. Shamsheer Daula, Dr. K.E Sreenivasa Murthy, G Amjad Khan," A Throughput Analysis on Page Replacement Algorithms in Cache Memory Management", International Journal of Engineering Research and applications (IJERA) ISSN: 2248- 9622 www.ijera.com Vol. 2, Issue 2, Mar-Apr 2012, Pp.126-130.

**[5]**Muthukumar, S. and P.K. Jawahar," HIT RATE MAXIMIZATION BY LOGICAL CACHE PARTITIONING IN A MULTI-CORE ENVIRONMENT", Journal of Computer Science 10(3): 492-498, 2014 ISSN: 1549-3636© 2014 Science Publicationsdoi:10.3844/jcssp.2014.492.498 Published Online 10 (3) 2014

**[6]** Yogesh Niranjan," Design and Implementation of Page Replacement Algorithm for Web Proxy Caching", Int.J.Computer Technology & Applications, Vol 4 (2), 221-225 IJCTA | Mar-Apr 2013 Available online@www.ijcta.com 221 ISSN: 2229-6093

**[7]**Ms. Richa Gupta and Dr. Sanjiv Tokekar," A Novel Pair of Replacement Algorithms on L1 and L2 Cache For FFT", International Journal on Computer Science And Engineering Vol.2 (1), 2010, 92-97

**[8]** ALFRED V. AHO and PETER J. DENNING AND JEFFREY D. ULLMAN," Principles of Optimal Page Replacement", Journal of the .Association for Computing Machinery, Vol. 18, No. 1, January 1971, pp. 80-93

**[9]** ww.cs.princeton.edu/courses/archive/fall10/cos318/

**[10]**www.en.wikipedia.org/wiki/Page_replacement_algorithm.

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

**[12]** Andrew S. Tanenbaum and Albert S. Woodhull, Operating Systems: Design and Implementation, Third Edition, Prentice Hall, 2006.