International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 10
June 2016

nLRU Page Replacement Algorithm

Mr. Puneeth Lakshmana Sankadal & Mrs. Sheela N*

*(Assistant professor Computer science department)

College: Sri Jayachamarajendra college of engineering, Mysore-570006

Abstract

nLRU is a page replacement technique where the replacement technique works exactly as LRU (least recently used) algorithm but keeping next n next pages in consideration before selecting the page to be replaced**.** Preferably n would be the number of frames in the main memory.

In proposed replacement algorithm the least recently used page is selected to replace when replacement has to be occurred.

While selecting the least recently used page care is taken such that: **i)**The least recently used page is selected to replace if that page is not again used in next **n** number of pages.

Keywords:: Page faults, Frames, n, Least recently used page, Hit, Miss

1.Introduction:

We use a two – level memory hierarchy consisting of a faster but costlier main memory and a slower but cheaper secondary memory. Virtual memory systems use this hierarchy to bring parts of a Program into main memory from the secondary memory in terms of units called as pages. Pages is brought into main memory only when the executing process demands them; this is known as demand paging. A page fault is said to occur when a requested page is not in main memory and needs to be brought from secondary memory. In such a case an existing page needs to be discarded. The selection of such a page is performed by page replacement algorithms which try to minimize the page fault rate at the least overhead.

The Least Recently Used replacement policy selects that page for replacement which has not been referenced for the longest time. For a long time, LRU was considered to be the most optimum online policy. The problem with this approach is the difficulty in implementation. One approach would be to tag each page with the time of its last reference; this would have to be done at each memory reference, both instruction and data. LRU policy does nearly as well as an optimal policy, but it is difficult to implement and imposes significant overhead.

Sometimes might be the least recently used page may be the requested by the processor after few pages are used. As the previous LRU algorithm blindly uses the least recently used page for replacement and again when processor request for the same page very soon after the page is replaced resulting there are many page faults occurring during this process.

The introduced nLRU selects the page to be replaced such tat:

i)The most least recently used page is selected to replace if that page is not again used in next **n** number of pages.

ii) If the most least recently used page is available in next **n** frame then the second most least recently used page is used to replace and is again compared with same n number pages.

iii) On the whole the n least recently used pages are compared with **n** next pages when there are any page replacement are to be performed.

The **n** least recently used pages are kept track in an single dimensional array. The array is constantly being updated about the used pages and ranked from 1st least recently used to last least recently used.(1st LRU page is that

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 10
June 2016

page used very long ago and last LRU page is that page used while after 1st LRU and so on).

Further *part of the paper contains the literature survey followed by the methodology to solve the problem and followed by the test result and the comparison between the traditional methods and lastly the references.*

2. Literature survey:

The traditional methods which include LRU and Optimal page replacement algorithm is the backbone of the nLRU page replacement technique.

As optimal algorithm checks for last reference of the pages for to replace the newly requested page by the processor ,and if there are many pages to be requested by the processor in future the time to search for that pages is very large and we need to restrict that number of comparison and must select the page which may not be used for long time again to replace with the newly requested page by the processor.

As LRU technique blindly selects the least recently used page to replace the newly requested page, and where after long time processor need that page again where we have replaced that very recently. So care is to be taken that that which we are selecting must not be requested by the processor again very soon.

Considering these 2 drawbacks the introduced algorithm selects that page to be replaced which is not requested by the processor again very soon and that exact number of pages to check for the future reference.

So as to select the page which might not be used by the processor for long time we are selecting the least recently used pages and comparing those pages with the next few pages. The next few pages can be decided by the programmer or by default it can be set to the number of frames used in that particular algorithm.

3. Methodology:

As discussed earlier the algorithm exactly works as same as LRU. The only thing to be discussed is how exactly n is selected and n future pages are compared when a page is to be replaced. What when comparison fails or succeeded.

Consider the below mentioned cases with 3 frames and hence n=3.

Case (i):

When there is a miss and there are empty frames the newly arrived page is directly placed into the frame.`

3	7

6	6
3	3
	7

As there are empty frames in the main memory the newly arrived pages can be directly placed into the main memory.

Assume 6, 3 were already present in the main memory and processor request page 7. As the other frame was free the newly requested pages in inserted into that location.

Case(ii):

When there is a hit we do not want to replace any pages as the available page is already available in the main memory.

7	5
5	5
6	6
7	7

Assume the newly requested page by the processor is 5 and when that pages is already present in the main memory there is no need to replace any of the pages from the frame as the requested page is present in the main memory.

Case(iii):

When there is a miss and there are no free frames in the main memory ,Here we need to compare the least recently used element with the next 3 pages which the process will request further.

Assume when the least recently used page is to be requested by the processor further (in the next 3 pages) then that page from the frames has to be retained from replacement.

To retain that page need to consider the below mentioned cases:

Case(iii.i):

If the least recently used pages is not further (in next 3 pages) requested by the process then that least recently used pages is used to replace the newly requested page.

7	6	4	2	9
7				
0				
1				

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 10
June 2016

Assume 0,7,1 is the most least recently used pages respectively as of now and hence as 0 is the most least recently used page 0 is compared with next 3 frames weather this pages is being requested by the processor or not.

As 0 is not present in 1of the next 3 frames then 0 is used to replace with requested frames.

The frames are updated as below:

7	6	4	2	9
7	7			
0	6			
1	1			

Case(iii.ii):

If the least recently used pages is further (in next 3 pages) requested by the process then that least recently used pages is retained in frames for the further execution

If the 1st least recently used page is to be requested by the processor further then that page from the frames has to be retained from replacement.

7	6	4	0	9
7				
0				
1				

Assume 0,7,1 is the most least recently used element respectively as of now and hence as 0 is the most least recently used and 0 is compared with next 3 frames weather this pages is being requested by the processor further.

As 0 is present in 1 of the next 3 frames then 0 is again updated in least recently used array i.e 7,1,0 will be updated into the most least recently used array respectively.

Now as 7 is the next least recently used page that page is compared with next 3 frames. As 7 is not part of the next 3 frames the presently requested pages is replaced with 7.

The frames are updated as below:

7	6	4	0	9
7	6			
0	0			
1	1			

Case(iii.iii):

Assume if both 1st least recently used and the 2nd least recently used pages are requested by the processor further then it is good practise that if we retain both the pages and replace with the 3rd least recently used page.

6	7	4	0	7
6	7			
0	0			
1	1			

Assume 0,7,1 is the least recently used pages respectively, Hence we 1st check for 0 weather it is requested by the processor in next 3 frames. As it is requested we now update least recently used array and now the array will be 7,1,0.Now we check for 7 ,as 7 is also being requested by the processor we go with last least recently used i.e the page which is recently used and that is 1.As 1 is not being requested by the processor in next 3 pages we retain 0,7 and use that frame with page 1 to replace 4.

The update frames are as below:

6	7	4	0	7
6	7	7		
0	0	0		
1	1	4		

With this we are able to reduce the 2 occurring page faults for next 2 pages as we have successfully retained the 2 pages.

Case(iii.iv):

Assume there are situations where all the 3 least recently used pages are being requested by the processor then we need to follow the tradition LRU technique where least recently used page is selected to replace newly requested page.

4. Results:

The below worked problem helps to know how exactly we reduce the number faults compared to LRU algorithm.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 10
June 2016

7	7	7	2	2	2	2	4	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	0	1
F	**F**	**F**	**F**	**H**	**F**	**H**	**F**	**F**	**H**	**H**	**H**	**H**	**F**	**H**	**H**	**H**	**F**	**H**	**H**

H refers to hit **F** refers to faults
Total number page faults are = 9.

The below table show step by step how every page is replaced:

1. Inserted into the empty frame.

		7

2. Inserted into the empty frame.

	7	0

3. Inserted into the empty frame.

7	0	1

4. As 7 is most least used page 7 is checked with next 3 pages. As 7 is not present 2 is replaced with 7.

0	1	2

6. As 1 is most least used page 1 is checked with next 3 page. As 1 is not present 1 is replaced with 3.

2	0	3

7. 0 is already present in the main memory it is a hit.

2	3	0

8. As 2,3,0 is the most least recently used page and all 3 are requested by the processor the algorithm works like the Traditional LRU. Hence replaced with 2.

3	0	4

9. As 3 is most least used page 3 is checked with next 3 pages. As 3 is present the next least recently used page i.e0 is compared with next 3 frames .As 0 is also requested by the processor now 4 compared with next 3 pages. As 4 is not present in the next 3 pages 2 is replaced with 4.

11. 0 is already present in the main memory it is a hit.

2	3	0

12. 3 is already present in the main memory it is a hit.

2	0	3

13. 2 is already present in the main memory it is a hit.

3	0	2

14. As 3 is the most least recently used and 3 is not present in next 3 pages 1 is replaced with 3.

0	2	1

16. 0 is already present in the main memory it is a hit.

0	2	1

17. 1 is already present in the main memory it is a hit.

0	2	1

18. As 0 is the most least recently used and 0 is present in next 3 pages the next least used page is compared with next 3 pages. As 2 is not present 7 is replaced with 2

0	1	7

19. 0 is already present in the main memory it is a hit.

0	1	7

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 10
June 2016

	3 | 0 | 2		
5. 0 is already present in the main memory it is a hit. 1 | 2 | 0	**10.** 3 is already present in the main memory it is a hit. 0 | 2 | 3	**15.** 2 is already present in the main memory it is a hit. 0 | 2 | 1 .	**20.** 1 is already present in the main memory it is a hit. 0 | 1 | 7

**** above shown array inside the table represent the Least recently used pages respectively.**

Result of Traditional LRU algorithm:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
F	**F**	**F**	**F**	**H**	**F**	**H**	**F**	**F**	**F**	**F**	**H**	**H**	**F**	**H**	**F**	**H**	**F**	**H**	**H**

H refers to hit **F** refers to faults

Total number page faults are = 12.

5. Algorithm:

Step1: Start
Step2: Read The string pattern
Step3: Read the number of frames
Step4: Assign 1st string to 1st result matrix(Compulsory miss)
Step5: For i=0 to N
If string[i]== result[i-1]
Retain the same resultant col(hit)
Else
Select the LRU page which is not in the next **n** page(miss)
Fault++
Step6: Stop

From the above mentioned example we have got 9 page faults and very less when compared to the traditional LRU technique where we get 12 page faults.
Number of page faults when compared to other page replacement technique are:

1.LRU=12 faults
2.FIFO=15 faults
3.Optimal=9 faults

5. Conclusion

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 10
June 2016

6. **References**

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

[2]. Mukesh Kumari et al, Int.J.Computer Technology & Applications,Vol 4 (4),683-685
SIMULATION OF LRU PAGE REPLACEMENT ALGORITHM FOR IMPROVING PERFORMANCE OF SYSTEM

[3]. PETER BAER GALVIN 7th edition text book for operating systems

[4].https://www.google.co.in/?gfe_rd=cr&ei=UPlZLmHJqLM8gfnrIGoCQ#q=page+replacement+algorithms+wiki