Least Recently Used Page Replacement using Last Use Distance (LRUL)

Ruchin Gupta
IT, Ajay Kumar Garg Engg. College, Ghaziabad, India

Narendra Teotia
IT, Ajay Kumar Garg Engg. College, Ghaziabad, India

ABSTRACT

Virtual memory technique is used in modern OS which permits the execution of a program while it is partially available in memory thus providing an illusion of very large memory to the user and freeing the user from the concern of large program size. This uses a page replacement technique such as first in first out (FIFO), least recently used (LRU), optimal etc. to replace a page in memory when a frame is needed and no free frame is available in memory. Here page replacement policy severely affects the performance of virtual memory. It has been observed that LRU approximates optimal so LRU and its variants are quite common in use in operating system as a reasonable choice of page replacement algorithm. But LRU performs very poor by generating continuous page faults while accessing looping pattern if all the pages does not fit in the memory simultaneously. This paper presents a simple modified LRU algorithm called LRUL overcoming this problem using the concept of last use distance (LUD). It is observed that the new algorithm gives better results than LRU.

General Terms

Operating system, page replacement algorithm.

Keywords

Operating system, LRU, page replacement algorithm.

1. INTRODUCTION

Operating system uses the concept of virtual memory to provide an illusion to a user having a very large amount of main memory available and allowing him/her to execute a program to be partially there in main memory [1]. In most of the operating systems virtual memory in implemented by demand paging. There are several advantages of using this concept like less I/O, efficient resource utilization etc. Performance of virtual memory is affected by the choice of page replacement technique used.

There are several page replacement techniques suggested by different researchers. Elizabeth J. O'Neil1, Patrick E. O'Neil1, Gerhard Weikum uses LRU-K page replacement algorithm or database disk buffering [2]. Sedigheh Khajoueinejad, Mojtaba Sabeghi, Azam Sadeghzadeh used fuzzy cache replacement policy [3].

Optimal technique is proven to be the best but it can not be implemented because it requires future knowledge of the reference string. Least recently used policy approximates it and that is the reason different variations of LRU have been implemented. It is well known however that there are many situations where LRU behaves far from optimal [4]. Under LRU an allocated memory page of a program will become a replacement candidate if the page has not been accessed for a certain period of time under two conditions: (1) the program does not need to access the page; and (2) the program is conducting page faults (a sleeping process) so that it is not able to access the page although it might have done so without the page faults. However, LRU page replacement implementations do not discriminate between two types of LRU pages and treat them equally [5]. So it means that LRU can be improved. Some page replacement policies and some other definitions are given below.

1.1 First in, First out (FIFO) method

In this when a page is needed, the page that has been in memory for the longest period of time is selected for replacement. The rationale is that a page that has only recently been swapped in will have a higher probability of being used again soon according to temporal locality principle.

1.2 Least Recently Used (LRU) method

Least recently used page replacement policy assumes that page reference pattern in the recent past is a mirror of the pattern in the near future. Pages that are accessed recently are likely to continue to be accessed and ought to be kept in physical memory.

1.3 Optimal method

This policy selects a page for replacement which will be used after the longest period of time. Since this requires knowledge of the future reference string,that’s why can not be implemented in the system. Hence this policy is used for comparative study only.

1.4 LRU k method

LRU k method has been successfully applied database disk buffering and has been shown to be better than LRU. Also LRU 2 has been found to be best for k=2. As per this policy LRU is a special case of LRU k where k=1.

The LRU-K Algorithm specifies a page replacement policy when a memory frame is needed for a new page requested: the page p to be dropped (i.e., selected as a replacement victim) is the one whose backward K-distance (bt(p,K)) is the maximum of all pages in memory. The only time the choice is ambiguous is when more than one page has $bt(p,K) = \infty$.In this case, a subsidiary policy may be used to select a replacement victim among the pages with infinite Backward K-distance; for example, classical LRU could be employed as a subsidiary policy.

2. MEMORY ACCESS PATTERN CLASSIFICATION

It is possible to draw following observations regarding memory access patterns.

Few no of pages are accessed very heavily and have a very high frequency of occurrences. It is also observed that few pages are accessed two or more times and are not used for a long time .Such occurrences of pages show high degree of temporal locality as temporal locality says that a recently accessed page will be accessed soon in future. Such kinds of accesses are also known as correlated accesses.

Some pages have many correlated accesses in a short period of time and then it is not accessed for a long time. Such kind of access pattern is known as scan accesses. Scan is performed on a large number of pages in relatively short time.

Few set of pages are accessed repeatedly like in looping pattern. This is known as loop pattern. The loop pattern is a LRU unfriendly pattern. In the worst case scenario, where loop accesses more pages than fitted into memory, LRU will have zero hit rate and page fault happens every time the next page in loop is accessed.

3. SUGGESTED ALGORITHM (LRUL)

Suggested algorithm uses last use distance (LUD) which is the distance between the current occurrence of a page to its last occurrence looking backward in the reference string from the current reference.

1. If free frame is available in memory then allocate the frame.
2. If no free frame is available, then find the last occurrence of the newly requested page back in the reference string and compute the distance between its last occurrence and it's recent occurrence (call it last use distance in short "LUD"), also calculate the LUD for the page which has come just before this newly requested page .Now if the LUD for these two pages come to be equal then replace the most recently used page, otherwise replace the LRU page from the memory. If there is no occurrence of the newly requested page or previous page then set its LUD to ∞.

4. EXAMPLE

In the following example; page faults are calculated using LRU, LRU2, LRUL, Optimal policy for the reference string 1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4.Results show that LRUL performs better than LRU and LRU2 giving results close to optimal incase of loop pattern.

Here LRU/Optimal, LRU2/Optimal, LRUL/Optimal defines a new parameter called page fault ratio (PFR). This PFR compares the performance of a policy compared to Optimal. More close the PFR value to 1 means better is the policy. Hence for a good policy PFR should decrease rapidly with increased no of frames.

It can be observed that PFR for LRUL is less and reduces rapidly towards 1 than others. Figure1 shows the plot between no. of frames and no. of page faults for all the 4 policies. Figure1 shows the effectiveness of the suggested algorithm in comparison to others and indicates its closeness to optimal policy.

Table 1. No of page faults using LRUL and comparison with others

No. of frames	LRU (series1)	LRU2 (series2)	LRUL (series3)	Optimal (series4)	LRUL /Optimal	LRU2/ Optimal	LRUL /Optimal
1	16	16	16	16	1	1	1
2	16	15	13	12	1.333333	1.25	1.083333
3	16	14	9	8	2	1.75	1.125
4	4	4	4	4	1	1	1

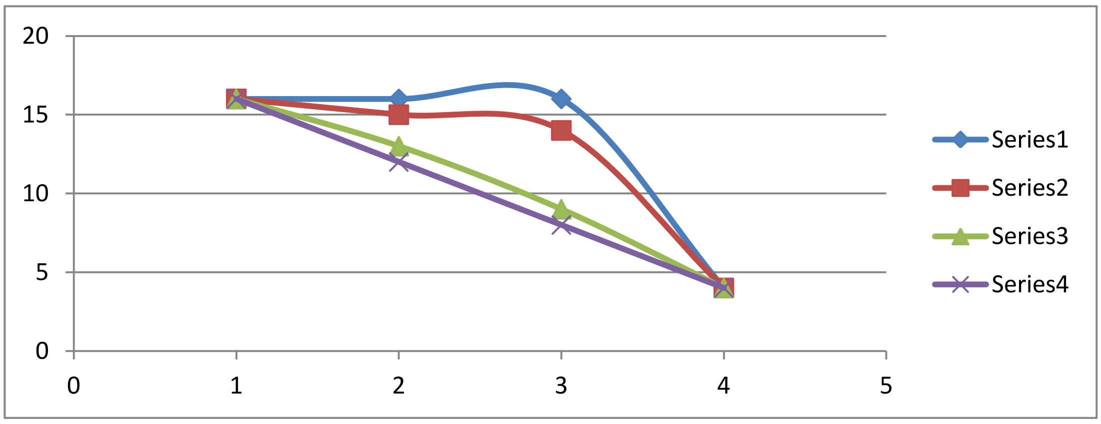

Fig 1: x axis =no. of frames and y axis= no. of page faults

5. CONCLUSIONS

It is observed that looping pattern accesses a set of pages repeatedly in a cyclic manner and if all these pages does not fit in the memory simultaneously then LRU generates continuous page faults and this is the worst behavior of LRU that's why such kind of patterns are known LRU unfriendly patterns. Suggested modified algorithm uses last use distance to determine a looping pattern and gives lesser no of page faults compared to LRU and LRU2. Also it gives results quite close to optimal. Simplicity is an important advantage of the suggested algorithm. In case of non loop pattern this simply reduces to LRU policy.

6. REFERENCES

[1] Abraham Silberschatz, Peter Baer, 1999.Operating System Concepts (5th Ed.).New York: John Wiley & Sons, Inc.

[2] Elizabeth J. O'Neil1, Patrick E. O'Neil1, Gerhard Weikum.The LRU-K Page Replacement Algorithm For Database Disk Buffering.SIGMOD Washington, DC, USA 1993 ACM.

[3] Sedigheh Khajoueinejad, Mojtaba Sabeghi, Azam Sadeghzadeh. A Fuzzy Cache Replacement Policy and its Experimental Performance Assessment, 2006 IEEE.

[4] Ben Juurlink, Approximating the Optimal Replacement Algorithm.CF’04,April 14–16, 2004, ACM 1581137419/ 04/0004.

[5] Song Jianga,, Xiaodong Zhangb. Token-ordered LRU: an effective page replacement policy and its implementation in Linux systems, 2004 Elsevier .