Journal of Engineering and Applied Sciences 13 (2): 478-483, 2018
ISSN: 1816-949X
© Medwell Journals, 2018

A Novel Approach to Improve LRU Page Replacement Algorithm

[1]Nabeel Zanoon, [1]Evon Abu-Taieh and [2]Hatem Salem Abu-Hamatta
[1]Faculty of Computer Information Systems, Al-Balqa Applied University, Aqaba, Jordan
[2]Faculty of Computer Information Systems, The University of Jordan, Aqaba, Jordan

Abstract: In the last decade's several page replacement algorithms had been implemented such as Least Recently Used (LRU), First Income First Out (FIFO) and optimal. Each of them has its own advantages and disadvantages. The process of page selection is time-consuming and depends on many factors: operating system and architecture as well as algorithms that will be used. Both LRU and optimal are implemented based on page fault rate. This study proposed LRU-Time dubbed as LRU-T which is a new algorithm to limits the number of pages the algorithm search through in order to reduce the page fault rate and tracking page frequencies during specified amount of time. The proposed algorithm was tested against the mentioned algorithms using a simulation program. The novelty behind the algorithm is improving the performance of LRU algorithm and increases its efficiency.

Key words: Replacement algorithm, LRU-time, optimal, page fault, locality, access pattern

INTRODUCTION

The process of replacing the page is important for operating systems whereas page replacement algorithms are the one that decides which page should be removed from the memory when a frame is not available. Some algorithms are developed to solve this issue these algorithms are different from each other according to the way they handle page removal process (Anthony, 2015). All page replacement algorithms are internally similar to the following: insertion, detection and searching for a page. All mentioned algorithms depend on specialized data structure where the performance of page replacement algorithm can be improved by using effective data structure (Kavar and Parmar, 2013). Most of the algorithm must handle huge amounts of data. The locality of references is considered as a shared attribute between programs. Hence, the majority of applications do not access all data at the same time. Instead of this, they reference only small part of data at of different point time (Shen *et al*., 2004). The locality of reference can be adopted in two different approaches: spatial locality of reference and temporal locality of references. The spatial locality that nearby memory locations will be referenced in the near future. Temporal locality depends on the time the page spent in the frame (O'neil *et al*., 1993). The locality of references can be increased by careful selection of data structure used in the algorithm. The data structure will reduce the page fault rate as well as the number of pages in working set. For example, a stack has high locality because the replacement algorithm of the stack always accesses the top. Some measure: memory reference, search speed and a total number of page touched are used to measure the performance of page replacement algorithm (Khulbe *et al*., 2014). LRU suffers from the following: first LRU has a linked list of pages. In the link list most recently used page at the front of the list and the least recently used page at the rear of the list. The linked list must be updated and maintained on every memory reference cycle. The search process for a page in the link list includes the calculation for all reference in the memory. Hence, reducing the performances of the overall algorithm LRU (Reddy, 2009). In this study, LRU-T was suggested reducing the scope of search and count fault page in that scope to improve search speed. LRU does not take the previous factors into account.

Frames allocation and locality of references: Memory is divided into a set of frames, frames are allocated for pages. The distribution strategy for each incoming process is of two types: equal allocation and proportional allocation. Equal allocation approach each incoming process gets the same number of frame (equal percentage). In proportional allocation approach here each process will get a proportional number of frames based on its size in comparison with the total size of all incoming processes (ChengJun, 2009).

The locality of process changes as it continues execution this behavior of typical process in execution, in its next phase of execution a process will reference a different set of pages and the number of page referenced may also be different the number of pages can decrease or

Corresponding Author: Nabeel Zanoon, Faculty of Computer Information Systems, Al-Balqa Applied University, Aqaba, Jordan

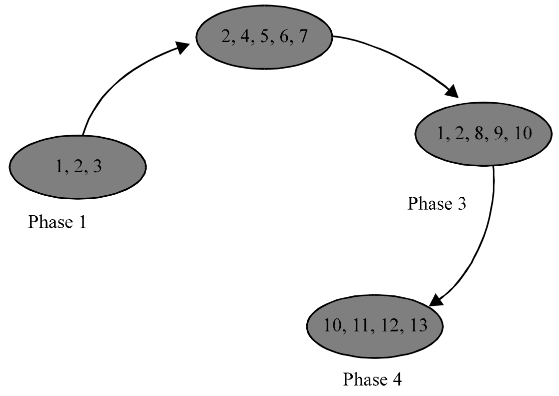


Fig. 1: Example of localities of a process during execution

increase with time, all this means that process will require a number o f frames allocated which varies with time, they are more frame allocated there will not beimprovement in its behavior if there are fewer frames allocated as shown in Fig. 1 (Garrido and Schlesinger, 2008).

The performance of paging algorithm depends on the distance string. The distance string depends on reference string and the paging algorithm (Tanenbaum and Bos, 2014). The stack distance affects the number of tests to match the current page which reflects negatively on the computing time according to Bennett and Kruskal, 1975).

LRU neglects the page use frequency. Hence, it is possible for a high-frequency page to be evicted. This makes LRU extremely vulnerable to scan access pattern. Furthermore, LRU does not work well with workloads that have clear large-scale access patterns until the pages belonging to the pattern fit to the main memory on the other hand, LRU works well with stack distances with common stationary distribution (Paajanen, 2007).

Page replacement evaluation metrics: The efficiency of page replacement algorithm is an open question for the study since there are several factors that effect on the algorithm performances (Brinkhoff, 2002). The memory-access has a clear impact on page replacement algorithm performances (Anthony, 2015). The main objective of page replacement algorithms was to reduce page fault rate which in turn enhance the effective access time of the memory. The effectiveness of the page replacement algorithm is based on the application behavior and memory access patterns (Rashidah *et al*., 2011). To analyze the performance of paging memory system some page trace experiments were conducted. A page trace is a sequence of Page Frame Numbers (PFNs) generated during the execution of a given program (Hwang and Jotwani, 2011). Several properties should cover the effective replacement algorithm, it should be able to distinguish between the hot and cold reside in cache to reduce the number of page faults. Furthermore, efficient algorithm implementation can be achieved if it uses a constant and small portion of memory that stores page history in the cache, it should not consume large memory space. Finally, the smart algorithm should be free from any assumptions that are not realistic and it should adopt references string dynamically to reduce of page faults (Dorrigiv *et al*., 2015).

Lru page replacement algorithm: The buffering algorithm in LRU has a flaw: LRU does not take into account the number of references to the same page. Hence, pages are all equal when evicted without noticing that one is more frequently used (Bagchi and Nygaard, 2004). LRU algorithm is based on a similar idea as to optimal by using the requests to elements to determine which elements to keep in the memory. LRU is usually implemented with a linked list. Therefore, it has a big drawback because moving elements to the most recently used position in the linked list at every request is expensive (O'neil *et al*., 1999). Neither LRU or optimal belong to a class of page-replacement algorithms called stack algorithms, a stack algorithm is one in which the pages kept in memory. Stack processing a trace that has a large number of distinct pages or a large average stack distance may require excessive computing time because of the number of tests (Coras *et al*., 2012). The LRU algorithm fails to handle the following three data access patterns:

- Each data block is only accessed once in a format of sequential scans
- For a cyclic (loop-like) data access pattern where the loop length is slightly larger than the buffer size, LRU always mistakenly evicts the blocks that will be accessed soon in the next loop
- In multiple streams of data accesses where each stream has its own probability for data re-accesses, LRU could not distinguish the probabilities among the streams (Wang, 2014)

One of LRU problems is looking for page in large scale of memory and it is necessary to maintain a linked list of all pages in memory. The algorithm searching takes extra time because the algorithm search for and update page table when page fault occurs (Bansal and Modha, 2004). In addition, frequently in pages is a problem since the LRU algorithm does not take into account frequency, and this is an indication that it does not work well with workloads that have wide-ranging access pattern (Jiang and Zhang, 2005). LRU can suffer from its pathological case when the working set size is larger than the cache and the application has a looping access pattern. In this case, LRU will replace all blocks before they are used again, resulting an increase in the MISS count (Juurlink, 2004).

Proposed approach (LRU-TIME): The policy implemented in LRU-T is based on limiting the pages to be searched hence, reducing the target page. LRU-T concentrates on frequency of the target page and time limit. Target page is chosen based on least frequently called, least recently used within the past time limit. The following presents the pseudo code of the suggested LRU-T:

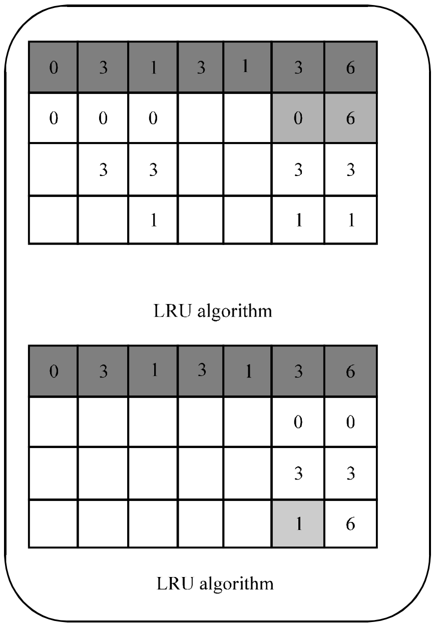
Algorithm:

```
$T_v = (F_n)$
INPUT  PAGES
FOR $P_{in}$TO$P_{in}$(n): P I+1
      IF  Frame = 0 THEN
                Replace page in frame
      Else                    // search of page in memory
FORTv (i)    TOTv (n) // interval of time
          IF $P_{in}$ == $P_m$ THEN
                Take the Next Page ($P_{in}$)
      Else ($P_{in}$ != $P_m$)  //Page fault
      Boolen Found = ($T_v$) [Last page and smallest frequently]
IF   Found THEN  Replace Page in Frame
V. P = $P_m$          // victim pages
Remove $P_m$
Replace $P_{in}$
End for
End for
```

The total memory size with respect to the number of page's effects the performances of LRU algorithm. Thus, the proposed approach came with the idea such that reducing updating scope rather than update overall page table. Furthermore, LRU-T takes into account the page reference frequency and last page in interval time.

In case of page fault, the proposed approach LRU-T finds and selects pages for removal that has least frequently and last within a specified interval of Time (T). LRU-T does not take into account which page to keep in the stack when searching as shown in Fig. 2. LRU-T assumes that all pages have same probability to reside and replaced in/from the memory based on specified time interval.

The algorithm policy (LRU-T) as in Fig. 2 shows that when page number 6 was called, the memory already had pages (3, 1 and 0) and page 0 was chosen to be evicted. LRU choose page 0, since, page 0 is the least recently used with no regard to how many times page 0 was called. On the other hand, LRU-T and when page 6 was called the evicted page is 1 because page 1 was the least frequency and last within time frame. LRU take the eviction decision based on the arrival time of the page while LRU-T takes into account arrival time and the frequency of the page calling within the specified interval of time.



0	3	1	3	1	3	6
0	0	0			0	6
	3	3			3	3
		1			1	1

LRU algorithm

0	3	1	3	1	3	6
					0	0
					3	3
					1	6

LRU algorithm

Fig. 2: Example for LRU policy and LRU-T policy

MATERIALS AND METHODS

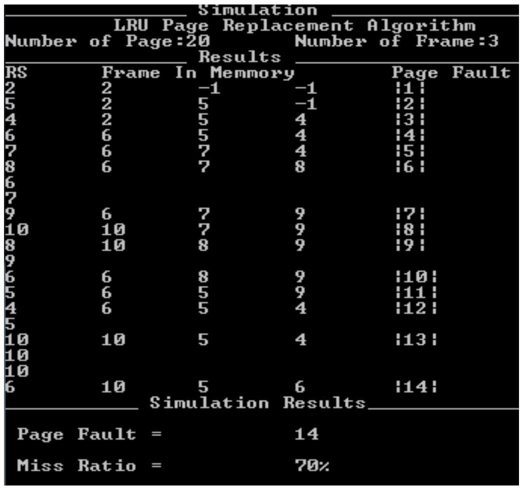
To evaluate the performance of the LRU-T, simulation program was written using VB.Net 2013. LRUT-T was compared with optimal and LRU by running a number of a sequence of memory references and counting page faults. The pages are input to the simulation program the simulator applies the policy of LRU-T, Opt and LRU and counts the page faults and the miss ratio. Hence, measuring the performance of each of the proposed algorithm (LRU-T) with LRU.

RESULTS AND DISCUSSION

In the experiment shown in Fig. 3, the LRU was applied on 20 pages with 3 frames. Notice that page number 4 is the targeted page to be replaced. The results are shown in Fig. 3, the number of fault pages reached 14 and the miss ratio (70%) but the suggested algorithm (LRU-T) that the number of fault pages reached 13 and the miss ratio (65%) as in Fig. 4.

In Fig. 4 notice that LRU-T replace page 4 with page 10, since, the stack had (5, 4 and 5). Page 4 is least frequently used and least recently used. The miss ratio is 65% since, as the number of pages increase the frequency increases and the stack distance increases also. The miss ratio will decrease as in the results next experiment.

Two experiments were conducted he Number of page's test were: 100, 200, 400, 600, 800 and 1000 in both experiments representing the cache size. The first

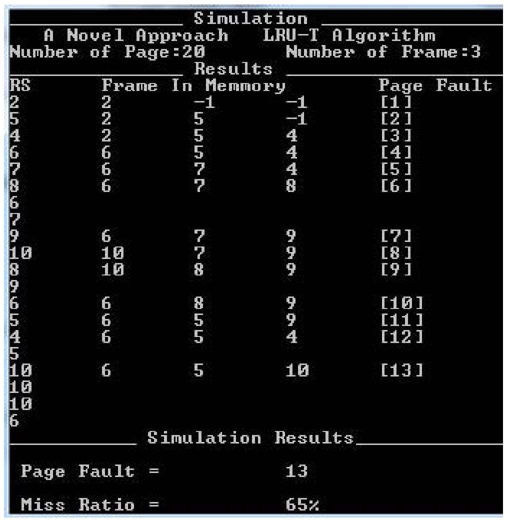


```
______________________ Simulation ______________________
          LRU Page Replacement Algorithm
Number of Page:20         Number of Frame:3
______________________ Results ________________________
RS          Frame In Memmory             Page Fault
2           2        -1        -1        |1|
5           2        5         -1        |2|
4           2        5         4         |3|
6           6        5         4         |4|
7           6        7         4         |5|
8           6        7         8         |6|
6
7
9           6        7         9         |7|
10          10       7         9         |8|
8           10       8         9         |9|
9
6           6        8         9         |10|
5           6        5         9         |11|
4           6        5         4         |12|
5
10          10       5         4         |13|
10
10
6           10       5         6         |14|
_______________ Simulation Results_____________________

 Page Fault =              14

 Miss Ratio =              70%
```

Fig. 3: Simulation results LRU with 20 pages and 3 frames



```
______________________ Simulation ______________________
   A Novel Approach   LRU-T Algorithm
Number of Page:20         Number of Frame:3
______________________ Results ________________________
RS          Frame In Memmory             Page Fault
2           2        -1        -1        [1]
5           2        5         -1        [2]
4           2        5         4         [3]
6           6        5         4         [4]
7           6        7         4         [5]
8           6        7         8         [6]
6
7
9           6        7         9         [7]
10          10       7         9         [8]
8           10       8         9         [9]
9
6           6        8         9         [10]
5           6        5         9         [11]
4           6        5         4         [12]
5
10          6        5         10        [13]
10
10
6
_______________ Simulation Results_____________________

 Page Fault =              13

 Miss Ratio =              65%
```

Fig. 4: LRU-T simulation results with 20 pages and 3 frames

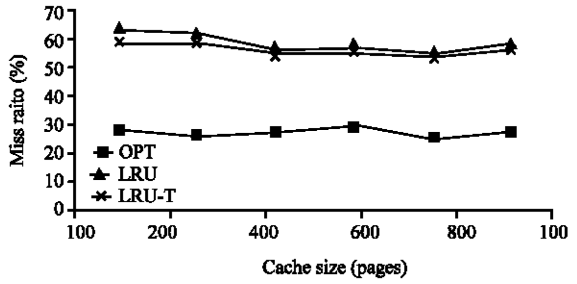


Fig. 5: Comparison of fault page rate in page replacement algorithms, 4 frames

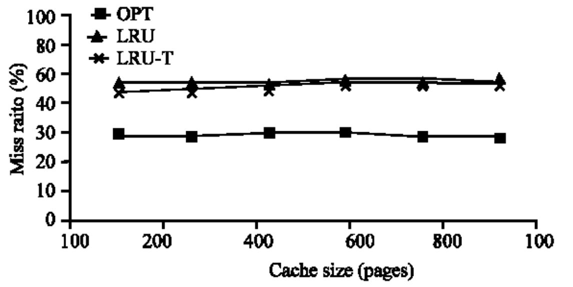


Fig. 6: Comparison of fault page rate in page replacement algorithms, 8 frames

experiment was conducted with 4 frames and the results shown in Fig. 5. As can be seen in the Fig. 5 the LRU-T outperformed LRU and optimal outperformed both. The miss ratio in optimal was 57% while LRU was 79% and LRU-T was 74% when the pages were 100. When the pages increase to 200, the optimal miss ratio was 56% and the LRU miss ratio was78% and LRU-T miss ratio was 75%. When the pages increase to 400, the optimal miss ratio was 56% and the LRU miss ratio was 78% and LRU-T miss ratio was 76%. When the pages increase to 600, the optimal miss ratio was 57% and the LRU miss ratio was 80% and LRU-T miss ratio was 79%. When the pages increase to 800, the optimal miss ratio was 56% and the LRU miss ratio was 79% and LRU-T miss ratio was 78%. When the pages increase to 1000, the optimal miss ratio was 55% and the LRU miss ratio was 79% and LRU-T miss ratio was 77%. Comparison of fault page rate in page replacement algorithms, 4 frames. In the second experiment, the number of page's test was: 100, 200, 400, 600, 800 and 1000 in experiment representing the cache size. Yet, the second experiment 8 frames were used as can be seen in Fig. 6 the LRU-T outperformed LRU and optimal outperformed both. The miss in optimal was 34% while LRU was 64% and LRU-T was 60% when the pages were 100. When the pages increase to 200, the optimal miss ratio was 32% and the LRU miss ratio was 63% and LRU-T miss ratio was 60%. When the pages increase to 400, the optimal miss ratio was 33% and the LRU miss ratio was 58% and LRU-T miss ratio was 57%. When the pages increase to 600, the optimal miss ratio was 35% and the LRU miss ratio was 59% and LRU-T miss ratio was 58%. When the pages increase to 800, the optimal miss ratio was 31% and the LRU miss ratio was 57% and LRU-T miss ratio was 57%. When the pages increase to 1000, the optimal miss ratio was 33% and the LRU miss ratio was 60% and LRU-T miss ratio was 59%. LRU-T and LRU were compared with simulation with pages ranged from 100-1000 as can be seen in Fig. 5 and 6; the first experiment was tested with 4 frames and the second with 8 frames. One can notice the improvement in certain parts.

CONCLUSION

The development of memory and operating systems requires an acceleration in the pages replacement, this paper proposed an approach to develop the (LRU) algorithm, in terms of the search range for the victim pages where the proposed approach works to narrow the search range by specifying a time period with the probability to survive and exit all the pages, so, that the page will be selected based on the least used (LRU) algorithm policy, the one that hasn't been used for a long time within the specific period which will leads to an increase in the speed rate of finding the victim page and helps to facilitate the execution process of the (LRU) algorithm. Based on the results, both number of page's fault and the memory access speed rate have been reduced which led the proposed approach (LRU-T) algorithm to advance in front of the (LRU) algorithm for the optimal algorithm.

REFERENCES

Anthony, R., 2015. Systems Programming: Designing and Developing Distributed Applications. Elsevier, Amsterdam, Netherlands, ISBN:9780128008171, Pages: 548.

Bagchi, S. and M. Nygaard, 2004. A Fuzzy Adaptive Algorithm for Fine Grained Cache Paging. In: Software and Compilers for Embedded Systems, Schepers, H. (Ed.). Springer, Berlin, Germany, ISBN:978-3-540-23035-9, pp: 200-213.

Bansal, S. and D.S. Modha, 2004. CAR: Clock with adaptive replacement. Proceedings of the 3rd USENIX Conference on File and Storage Technologies (FAST'04) Vol. 4, March 31, 2004, USENIX Association, San Francisco, California, pp: 187-200.

Bennett, B.T. and V.J. Kruskal, 1975. LRU stack processing. IBM. J. Res. Dev., 19: 353-357.

Brinkhoff, T., 2002. A Robust and Self-Tuning Page-Replacement Strategy for Spatial Database Systems. In: Advances in Database Technology, Jensen, C.S., S. Saltenis, K.G. Jeffery, J. Pokorny and E. Bertino et al. (Eds.). Springer, Berlin, Germany, ISBN:978-3-540-43324-8, pp: 241-276.

ChengJun, W., 2009. The research on the dynamic paging algorithm based on working set. Proceedings of the 2nd International Conference on Future Information Technology and Management Engineering (FITME'09), December 13-14, 2009, IEEE, Sanya, China, ISBN:978-1-4244-5339-9, pp: 396-399.

Coras, F., A. Cabellos-Aparicio and J. Domingo-Pascual, 2012. An Analytical Model for the LISP Cache Size. In: Networking, Bestak, R., L. Kencl, L.E. Li, J. Widmer and H. Yin (Eds.). Springer, Berlin, Germany, ISBN:978-3-642-30044-8, pp: 409-420.

Dorrigiv, R., M.R. Ehmsen and A. Lopez-Ortiz, 2015. Parameterized analysis of paging and list update algorithms. Algorithmica, 71: 330-353.

Garrido, J.M. and R. Schlesinger, 2008. Principles of Modern Operating Systems. Jones & Bartlett Learning, Burlington, Massachusetts, USA., ISBN-13:978-0-7637-3574-6, Pages: 489.

Hwang, K. and N. Jotwani, 2011. Advanced Computer Architecture, 3e. McGraw-Hill Education, New York, USA., ISBN:9789339220938, Pages: 723.

Jiang, S. and X. Zhang, 2005. Token-ordered LRU: An effective page replacement policy and its implementation in Linux systems. Perform. Eval., 60: 5-29.

Juurlink, B., 2004. Approximating the optimal replacement algorithm. Proceedings of the 1st Conference on Computing Frontiers, April 14-16, 2004, ACM, Ischia, Italy, ISBN:1-58113-741-9, pp: 313-319.

Kavar, C.C. and S.S. Parmar, 2013. Improve the performance of LRU page replacement algorithm using augmentation of data structure. Proceedings of the 4th International Conference on Computing, Communications and Networking Technologies (ICCCNT'13), July 4-6, 2013, IEEE, Tiruchengode, India, ISBN:978-1-4799-3926-8, pp: 1-5.

Khulbe, P., S. Kumar and N. Yadav, 2014. An assessment of hybrid LRU (H-LRU) with existing page replacement algorithms. Intl. J. Comput. Appl., 99: 51-53.

O'neil, E.J., P.E. O'Neil and G. Weikum, 1999. An optimality proof of the LRU-K page replacement algorithm. J. ACM., 46: 92-112.

O'neil, E.J., P.E. O'neil and G. Weikum, 1993. The LRU-K page replacement algorithm for database disk buffering. ACM. Sigmod Rec., 22: 297-306.

Paajanen, H., 2007. Page replacement in operating system memory management. Master's Thesis, University of Jyvaskyla, Jyvaskyla, Finland.

Rashidah, A., C.L. Gan and M.Y. Rubiah, 2011. Recovery of Memory Based on Page Replacement Algorithms. In: Informatics Engineering and Information Science, Abd-Manaf, A., A. Zeki, M. Zamani, S. Chuprat and E. El-Qawasmeh, (Eds.). Springer, Berlin, Germany, ISBN:978-3-642-25452-9, pp: 499-511.

Reddy, C.M.K., 2009. Operating Systems Made Easy. University Science Press, New Delhi, India, ISBN:978-81-318-0743-9, Pages: 103.

Shen, X., Y. Zhong and C. Ding, 2004. Locality phase prediction. ACM. Sigplan Not., 39: 165-176.

Tanenbaum, A.S. and H. Bos, 2014. Modern Operating Systems. 4th Edn., Prentice Hall, Upper Saddle River, New Jersey, USA., ISBN:9780133591620, Pages: 1101.

Wang, H., 2014. Study of page replacement algorithm based on experiment. Appl. Mech. Mater., 530: 895-898.