A Comparison of Page Replacement Algorithms in Linux Memory Management

Hasan M H Owda , Munam Ali Shah, Abuelgasim Ibrahim Musa, Manzoor Ilahi Tamimy
Department of Computer Science
COMSATS Institute of Information Technology
Islamabad, Pakistan
hasan_ouda{at}hotmail.com

Abstract— **The speed and competence of a processor depends on the time it takes in handling instructions. The speed of a processing device not only depends on its architectural features like chipset, transport buses and cache but also on the memory mapping scheme. A virtual memory system needs a page replacement algorithm to decide which pages should be evicted first as of the memory in case if a page fault occurs. Many page replacement algorithms have been designed and implemented where each algorithm attempts to reduce the page fault time while endure least amount of overhead. This paper surveys existing page replacement algorithms for Linux operating system and critically analyzes their advantages and design constraints.**

Keywords-Replacement Algorithms; Linux memory managent; LRU; ARC; CAR;

I. INTRODUCTION

Memory management is one of the most important components of any operating system. Multi-programming feature in any operating system could only be achieved if the memory management module has been efficiently designed. This is the main reason the hardware designers of computer systems deploy a two-level memory hierarchy consisting of main memory which is fast but expensive and secondary memory which is slow but cheaper.

Virtual memory (VM) is a technique which is used to compensate the shortage of Random Access Memory (RAM). VM, which could be deployed both by using software and hardware, moves the data as of RAM into the temporary space in hard disk called paging file. The partial shifting of the computer program into the main memory provides organizational structures [1]. Demand paging is a concept in operating systems that fetch only those pages as of VM into the main memory if the executing process demands them. A page fault occurs, when the requested page is not present in main memory, it is brought in from secondary memory. In this case, some threads need to govern these activities to avoid any page faults. It is the responsibility of page replacement algorithm to decide which pages needs swap in or swap out of the memory when memory needs to be reallocated.

This paper outlines the memory management techniques and critically analyzes page replacement algorithms for different operating system including Linux. The rest of the paper is organized as follow. In Section II, we discuss features of memory management systems. Section III, investigates advantages and disadvantages of some of the page replacement algorithms such as Belady's MIN, Least Recently Used (LRU), CLOCK, Dueling CLOCK, LRU-K, Low Inter-reference Recency Set (LIRS), CLOCK-Pro, Adaptive Replacement Cache (ARC) and Clock with Adaptive Replacement CAR. Section IV concludes the discussion.

II. MEMORY MANAGEMENT SYSTEM

Virtual memory is necessary for every operating system as it makes multi-programming convenient and possible. Firstly, it helps to separate tasks from each other by encapsulating them in their private address spaces. Secondly, virtual memory can offer operating systems' tasks and processes a persuasion of more memory available than is actually possible. And lastly, by using virtual memory, there might be various copies of the same program, linked to the same addresses, running in the system. There are at least two known mechanism to implement virtual memory: segmentation and paging [13].

The features of VM are not limited to segmentation and paging but it also provides features described below:

A. Huge address Space

It's actually above one operating system contains a large amount of memory as it appears in the system. Virtual memory system can be numerous times more than the actual memory [1, 2].

B. Protection

Each process has a virtual address space in its own system. Virtual address spaces and is completely separate from each

other and then run a single application can not affect each other. In addition, the areas of memory devices from the virtual memory system to protect against writing. These rogue applications from being overwritten by code and data security.

C. Balanced Physical Memory Allocation:

The memory management subsystem allows each running process in the system an equal share of physical memory.

D. Memory Mapping:

Is the use of memory address space of the process of mapping to map the files, Linked to the contents of the file directly into the virtual address space of the process.

E. Shared Virtual Memory:

A separate process virtual memory address space allows for some cases, this process requires memory to share. For example, in the bash shell command system range can be implemented. Several copies of the party, each process has a virtual address space, is only one copy in physical memory and run part of the process bash it would be better for all [2, 3,and 4].

III. PAGE REPLACEMENT ALGORITHIMS

In this section, we discuss page replacements algorithms. The emphasis is on highlighting the key features and design constrains of the discussed algorithms.

A. Least Recently Used (LRU)

It's kind of a type of algorithm used to manage memory. it is having an most favorable strategy .This strategy standard of district state as rejects the least lately used substance first for the program and data references, and it requires keeping path of what was used when substitute was occurred as well every page will be recognize by the time of its last position**,** but the problem of this substitute that it is tricky to be implement and impose a important overhead.

B. Belady's MIN

Belady's MIN serves as a theoretical optimum in performance as compared to other algorithms. Belady's MIN always discard pages will not be used for the longest time in the future *(page replacement algorithms)* .This result is referred as" Belady's MIN algorithm" and frequently called as "the clairvoyant algorithm". In Belady's MIN replacement decisions depends on the coming page sequence to decide which memory pages to page out but it is impossible to predict whether these pages will be needed in the future or not, Because of that this type of algorithm is unwieldy for the real systems. One can compare the effectiveness of the actually chosen of other replacement algorithm so it can be after the comparison useful in the simulation studies as it gives a lower bound on page fault rates under various operating circumstances.

C. CLOCK

In CLOCK, will remove a page that has not been referenced recently, as it keeps a circular list of pages in memory, as the entire page frames are conceive to be arranged in a match circular list. Each page has a reference bits and can find a page with that reference. There will be "hand" pointing to the last examined page frame in the list When a page fault occurs and no empty frames exist, then the R (referenced) bit is inspected at the hand's location. I.e. the oldest page is inspected. If the reference bit of the page is set, then the bit is reset to zero and the hand is advanced to point to the next oldest page. This process continues till a page with reference bit zero is found. This page is removed from the buffer and a new page is brought in its place with reference bit set to zero.

D. Dueling CLOCK

Dueling CLOCK is an adaptive replacement policy that is based on the CLOCK algorithm. A major disadvantage of the regular CLOCK algorithm is that it is not scan resistant i.e. it does not allow scanning to push frequently used pages out of main memory [8, 9]. To combat this disadvantage, pants are offered resistance to the survey and a set of algorithms and perspectives fencing dominant algorithm for adaptively altering the page moreover a Clock or resistance to the review are used to conclude pants's.

In order to create clock scan resistant, only desires to alter hands on the clock in its place of the oldest page, in the buffer should refer to the latest page. Now, during the scan sequence, and is inspected the same page every time frame, Stored in the frame buffer and page often for the use of the rest of the page to scan all the pages.

The Dueling CLOCK algorithm advocate a technique to adaptively utilize the above two algorithms, explicitly, CLOCK and scan resistant CLOCK. The cache is separated keen on three groups, G1, G2, and G3. Small group G1 for all time uses CLOCK algorithm for replacement while small group G2 forever uses the scan resistant CLOCK strategy. The superior group G3 can use also CLOCK or scan resistant CLOCK policies depending ahead the relation recital of groups G1 and G2. In order to settle on the substitution rule for G3, a 10 bit policy select counter (PSEL) is used. The PSEL counter is decremented whenever a cache miss occurs on the cache set in group G1 and the counter is incremented whenever a cache miss occurs on the cache set in group G2. Now, group G3 adopts CLOCK replacement policy when the MSB of PSEL is 1, other G3 uses the scan resistant CLOCK policy. In practice, Dueling CLOCK algorithm has been made known to make available considerable performance development in excess of LRU when applied to the problem of L2 cache management.

E. LRU-K

The LRU strategy at the same time as evicting pages and neglecting the frequency. LRU-K was proposition which evicts pages with the largest backward K-distance. Backward

K-distance of a page p is the distance rearward from the current time to the K^th most recent reference to the page p. Since we have to believe K^th most fresh reference to a page according to the strategy, it favoritism pages which are accessed regularly within a little time. The outcome illustrate that LRU-K perform superior than LRU [6, 12]; while senior K does not result in a considerable augment in the recital, but has high accomplishment transparency.

F. Low Inter-reference Recency Set (LIRS)

The Low Inter-reference Recency Set algorithm takes keen on consideration the Inter- Reference Recency of pages as the dominant factor for eviction [14]. The Inter-Reference Recency (IRR) of a page refers to the number of other pages accessed between two consecutive references to that page. It is assumed that if current IRR of a page is large, then the next IRR of the block is likely to be large again and hence the page is suitable for expulsion as per Belady's MIN. It needs to be well-known that the page with tall IRR preferred for expulsion may also have been newly used.

The algorithm distinguishes between pages with High IRR (HIR) and those with Low IRR (LIR). The number of LIR and HIR pages is chosen such that all LIR pages and only a small percentage of HIR pages are kept in cache. Now, in case of a cache miss, the resident HIR page with highest recency is removed from the cache and the requested page is brought in. Now, if the new IRR of the requested page is smaller than the recency of some LIR page, then their LIR/ HIR statuses are interchanged. Regularly only approximately 1% of the cache is used to stock up HIR pages whilst 99% of the cache is set aside for LIR pages [11].

G. CLOCK-Pro

The CLOCK-Pro algorithm tries to estimate LIRS using CLOCK. Use again distance, which is equivalent to IRR of LIRS, is a vital parameter for page substitute decision in CLOCK-Pro [4, 5, 8, and 10]. When a page is accessed, the reuse distance is the period of time in terms of the number of other distinct pages accessed since its last access. A page is categorized as a cold page if it has a large reuse distance or as a hot page if it has a small reuse distance.

Let the size of main memory be *m*. It is alienated keen on hot pages having size m_h and cold pages having size m_c. A part from these, at most *m* non-resident pages have their history access information cached. Hence a total of *2m* meta-data entries are present for keeping track of page access history. A single list is maintained to place all the accessed pages (either hot or cold) in the order of page access. Obviously, the pages through little recency are at the list head and the pages with big recency are at the list conclusion.

After a cold page is accepted into the list, a test period is granted to that page. This is done to give the cold pages a chance to compete with the hot pages. If the page is re-accessed during the test period, it is turned into a hot page. On the other hand, if the page is not re-accessed, then it is removed from the list. A cold page in its test period can be removed out of memory; however, its page meta-data is kept in the list for the test purpose until the end of its test period. The test period is set as the largest recency of the hot pages. The page entries are maintained in a circular list. For each page, there are three category bits; a cold/hot indicator (Fig. 1), a reference bit and for each cold page an indicator to determine if the page is in test period.

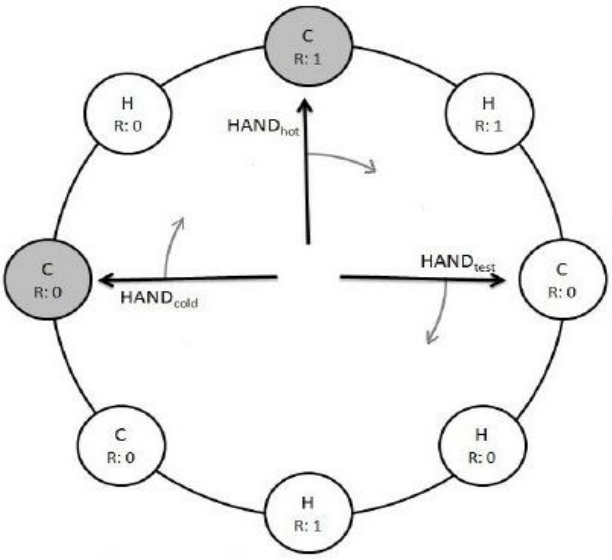

Figure 1. Working of CLOCK-Pro Algorithm

After a cold page is established keen on the list, a test phase is approved to that page. This is done to give the cold pages a chance to struggle with the hot pages. If the page is re-accessed for the duration of the test phase, it is twisted keen on a hot page. On the other hand, if the page is not re-accessed, then it is unconcerned from the list. A cold page in its test phase can be unconcerned out of memory; however, its page meta-data is reserved in the list for the test reason until the end of its test phase. The test phase is set as the major recency of the hot pages. The page entries are maintained in a spherical list. For each page, there are three status bits; a cold/hot pointer, a reference bit and for each cold page an pointer to settle on if the page is in test phase.

In CLOCK-Pro, there are three hands. HAND cold points to the last occupier cold page i.e., the cold page to be next replaced. Through page replacement, if the reference bit of the page pointed by HAND cold is 0, the page is dispossessed. If the page is in the test period, its meta-data will be saved in the list. If the reference bit is 1 and the page is in test phase, it is turned as a hot page and the reference bit is reset.

H. Adaptive Replacement Cache (ARC)

The Adaptive Replacement Cache (ARC) [11, 12] algorithm keeps a track of both commonly used and lately used pages, along with some of the past data as regards expulsion together.

ARC maintains two LRU lists: L1 and L2. The list L1 includes all the pages that have been accessed accurately once

lately, while the list L2 includes the pages that have been accessed at least twice lately. Thus L1 can be thinking of as capture short-term effectiveness (recency) and L2 can be thinking of as capture long phrase effectiveness (frequency). Every of these lists are rip into peak cache entry and base spirit entry.

That is, L1 is ripping keen on T1 and B1, and L2 is tearing hooked on T2 and B2. The entry in T1 union T2 constitutes the cache, while B1 and B2 are spirit lists. These spirit lists remain a track of lately dispossessed cache entry and assist in adapting the behavior of the algorithm. In addition, the ghost lists contain just the meta-data and not the real pages. The cache address list is thus prepared into four LRU lists as:

1. T1, for current cache entry
2. T2, for common entry, referenced at slightest twice
3. B1, spirit entry lately dispossessed as of the T1 cache, but are motionless tracked
4. B2, similar spirit entry, but dispossessed as of T2

If the cache size is c, then |T1 + T2| = c. assume |T1| = p, after that |T2| = c - p. The ARC algorithm frequently adapts the cost of parameter p depending on the present workload special treatment recency or regularity. If recency is more well-known in the existing workload, p will be increases; while if frequency is more prominent, p decreases (c - p increase).

In addition, the dimension of the cache index, |L1| + |L2| = 2c. For a fixed p, the algorithm for substitute would be as:

- If |T1| > p, restore LRU page within T1
- If |T1| < p, restore the LRU page within T2
- If |T1| = p and the miss page is within B1
- Restore the LRU page within T2
- If |T1| = p and the missed page is within B2.

The changed copy of the cost of p is fixed in the subsequent scheme:-

- If there is a strike during B1 then the information store as of the peak of vision of recency has been helpful and more breathing space should be selected to store up the smallest amount newly used one instance data.
- Therefore, we are supposed to lift up the dimension of T1 proposed for which the cost of p ought to rise.

If there is a hit in B2 then the data stored from the point of view of frequency was more relevant and more space should be selected to T2. Thus, the value of p should decrease.

The quantity by which p should diverge is known by the family member sizes of B1 and B2.

I. CLOCK with Adaptive Replacement (CAR)

CAR attempts to join the adaptive strategy of ARC by means of the accomplishment efficiency of CLOCK [7, 8]. The algorithm maintains four twice as linked lists T1, T2, B1, and B2. T1 and T2 are CLOCKs while B1 and B2 are easy LRU lists. The perception after these lists is equivalent as that for ARC. In addition, the lists T1 and T2 i.e. the pages in the supply, have a position bit that can be place or reset. Instinctively $T1^{\wedge}{}_0$, B1 point toward "recency" (Fig. 2).

The accurate description of four lists is as follows:

1. $T1^{\wedge}{}_0$ and B1 includes all the pages that are referenced accurately once since its most fresh deportation from T1 U T2 U B1 U B2 or was never referenced before since its commencement.

2. $T1^{\wedge}{}_1$, B2 and T2 includes all the pages that are referenced more than once seeing as its most recent deportation from T1 U T2 U B1 U B2.

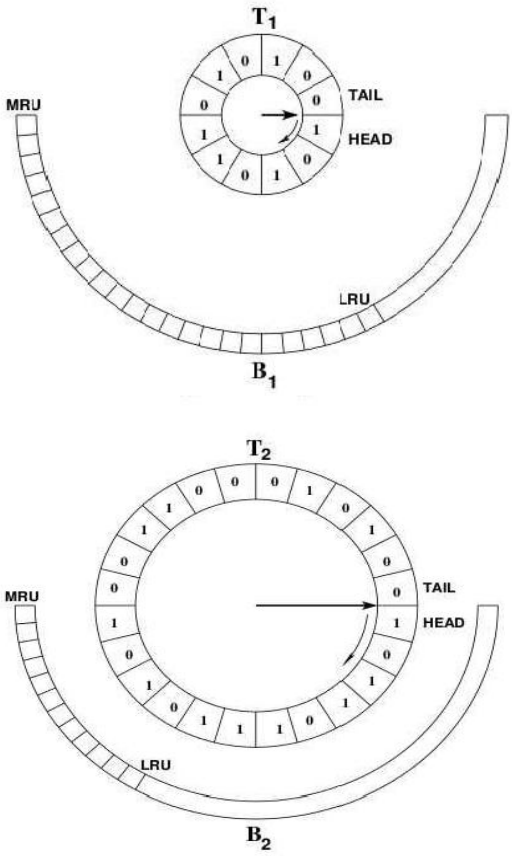

Figure. 2 A visual description of CAR

The most two essential constraint on the size of T1, T2, B1 and B2 are:

- $0 \leqslant |T1| + |B1| \leqslant c$. By description, T1 U B1 captures recency. The size of freshly accessed pages and regularly accessed pages remain on altering. This prevent pages which are accessed only once as of attractive up the intact cache address list of size 2c since growing size of T1 U B1 indicate that the freshly referenced pages are not being referenced again which in turn means the recency data that is stored is not obliging. Thus it means that only the

regularly used pages are re-referenced or new pages are being referenced.

- $0 \leqslant |T2| + |B2| \leqslant 2c$. If only a set of pages are being accessed frequently, there are no new references. The cache directory has information as regards only frequency.

The scheme behind the algorithm is as follows. In case of a cache hit, the requested page is delivering from the cache and its location bit is set. Instead, if the page is not in the cache but is present in list B1, this indicates the page had been used once recently before being evicted from the cache. This page is then moved to the head of T2 and its reference bit is set to 0. Also, a hit in B1 indicates that pages used once recently are required again implying a recency favoring workload. Hence the value of p has to be improved resulting increase in the size of T1.

Finally if the page is not found in B1 U B2, then the page is added to the MRU position in T1. In any of the above cases if the cache is full ($|T1| + |T2| = c$), then the CLOCK policy is applied on either T1 or T2 depending on the parameter p.

IV. CONCLUSION

Current computers often have some form of virtual memory. In the simplest form, each process' deal with gap is separated up into unvarying sized blocks called pages, which can be located into any existing page frame in main memory. There are many page substitution algorithms. LRU-based on working set size evaluation is an effectual method to carry memory resource management.CAR and ARC Entertainment and Tab solve the problem of cache hits. CLOCK performance of the adaptive control of the principles of efficiency, try to join ARC. The landscape is very self-tuning of the car's load is pre-existing information is not for use in this environment stand out. There is also a secondary resistance check. This paper outlines that enhancement of page substitution algorithms guide to professional use of main memory. These techniques one by one are more appropriate by extensively reducing its overhead. Investigational evaluation shows that, these algorithms are capable of dropping the overhead with adequate exactitude to develop memory allotment decision.

REFERENCES

[1] L. A. Belady, ―A study of replacement algorithms for virtual storage computers,‖ *IBM Sys. J., vol. 5, no. 2*, pp. 78–101, 1966.

[2] M. J. Bach, The Design of the UNIX Operating System. Englewood Cliffs, NJ: Prentice-Hall, 1986.

[3] A. S. Tanenbaum and A. S. Woodhull, Operating Systems: Design and Implementation. Prentice-Hall, 1997.

[4] A. Silberschatz and P. B. Galvin, Operating System Concepts.Reading, MA: Addison-Wesley, 1995.

[5] J. E. G. Coffman and P. J. Denning, Operating Systems Theory. Englewood Cliffs, NJ: Prentice-Hall, 1973.

[6] M. K. McKusick, K. Bostic, M. J. Karels, and J. S. Quarter man, The Design and Implementation of the 4.4BSD Operating System. Addison-Wesley, 1996.

[7] 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.

[8] 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.

[9] S. Jiang, and X. Zhang, ―LIRS: An Efficient Policy to improve Buffer Cache Performance‖, *IEEE Transactions on Computers*, pp. 939-952, 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 Tecchnical Conference*, pp. 35, 2005.

[11] N. Meigiddo, and D. S. Modha, ―ARC: A Self-Tuning, Low overhead Replacement Cache‖, *IEEE Transactions on Computers*, pp. 58-65, 2004.

[12] 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.

[13] 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.

[14] S. Jiang and X. Zhang, ―LIRS: An efficient low inter-reference recency set replacement policy to improve buffer cache performance, ‖ in Proc. ACM SIGMETRICS Conf., 2009.