Improving the Super-Paging Swapping Algorithm

Yair Wiseman
Computer Science Department
Bar-Ilan University
Ramat-Gan 52900
Israel
wiseman@cs.biu.ac.il

Abstract

The contemporary algorithms used by paging swapping systems are quite old and they usually do not take into account the possibility of super-paging. We suggest incorporating an uncommon page replacement algorithm so as to enhance the traditional swapping algorithms. This enhanced approach can efficiently handle the different size of pages and achieve better results.

Keywords: Super-Pages, Virtual Memory, Page Replacement Algorithms.

1. Background

Recently, a new page replacement policy has been proposed by Megiddo and Modha named ARC (Stands for Adaptive Replacement Cache) [1,2,3]. The new policy outperforms the traditional LRU used by many paging systems. Actually, the authors claim their policy outperforms also the LRU of many other systems e.g. web browsers or RIAD controller.

The main concept of ARC is to have two lists of active pages (one for the frequently used pages and one for the most recent pages) and to endow the list that is performing the best with a larger memory space. ARC is about 10%-15% more time consuming than LRU, but the hit ratio is in average about as twice as LRU. Small space overhead, however, is needed. The main advantage over the “Least Frequently Used” (LFU) [4] policy is that stale pages do not retain in the memory.

This paper inquires whether the ARC can also outperform the LRU when using SuperPaging systems [5]. SuperPages are larger pages that are pointed by the TLB. The internal memory of modern computers has been significantly increased during the last decade. However, the TLB coverage (i.e. the size of the memory that can be pointed
directly by the TLB) has been increased by a much lower factor during the same period [6,7]. Therefore, several new architectures like Itanium, MIPS R4x00, Alpha, SPARC and HP PA RISC support multiple page size of the frames pointed by the TLB. In that way the memory size pointed directly by the TLB is higher and the overhead of the page table access time is reduced. In addition, many modern operating systems support superpaging e.g. some versions of Linux [8] or HP-UX [9]

2. The ARC Algorithm

ARC is defined as follow: Let \( c \) be the number of pages in the memory and let \( L_1 \) and \( L_2 \) be two linked lists. \( L_1 \) contains the pages that have been accessed just once, while \( L_2 \) contains the pages that have been accessed at least twice. The allowed operations on \( L_1 \) and \( L_2 \) are the same operations that are allowed on an LRU linked list. \(|L_1|+|L_2|\) is a not negative number and not bigger than \( 2c \). \( L_1 \) is a not negative number and not bigger than \( c \), but \( L_2 \) can be bigger than \( c \).

When a page is accessed; if the page is in \( L_1 \) or \( L_2 \), it will be moved to the MRU of \( L_2 \); otherwise it will be moved to the MRU of \( L_1 \). If adding the new page makes \(|L_1|+|L_2| > 2c \) or \(|L_1| > c \), then if \( L_1 \) (before the addition) contains less than \( c \) pages, the LRU of \( L_2 \) will be taken out; otherwise the LRU of \( L_1 \) will be taken out.

\(|L_1|+|L_2|\) is not bigger than \( 2c \), but the size of the memory is just \( c \). This leads to the following partition: Let \( T_1 \) be the most recent pages in the memory and \( B_1 \) be the least recent pages in the memory. Similarly, \( L_2 \) is partitioned into \( T_2 \) and \( B_2 \). Therefore, \( T_1 \) and \( T_2 \) contain the pages that are actually in the memory. When a page is moved from a "T" list to a "B" list, it will be taken out of the memory.

Let the threshold \( p \) be the current target size for the list \( T_1 \).

- If \(|T_1| > p\), move the LRU of \( T_1 \) to be the MRU of \( B_1 \).
- If \(|T_1| < p\), move the LRU of \( T_2 \) to be the MRU of \( B_2 \).
- If \(|T_1| = p\), if the accessed page has been in \( B_2 \), move the LRU of \( T_1 \) to be the MRU of \( B_1 \). (Because \( p \) is going to be decremented). If the accessed page has been in \( B_1 \) or has not been in the memory, move the LRU of \( T_2 \) to be the MRU of \( B_2 \).
If there is a hit in $T_1$ or $T_2$, do nothing.

If there is a hit in $B_1$, if the size of $B_1$ is at least the size of $B_2$, increment the threshold $p$ by 1; otherwise, increment $p$ by $|B_2| - |B_1|$.

If there is a hit in $B_2$, if the size of $B_2$ is at least the size of $B_1$, decrement the threshold $p$ by 1; otherwise, decrement $p$ by $|B_1| - |B_2|$.

The increments and the decrements are subject to the stipulation $0 \leq p \leq c$.

ARC is better than LRU because if a process scans a large database, there will be no hits; hence the threshold $p$ will not be modified and the pages in $T_2$ will remain in the memory. ARC is better than LFU, because stale pages do not remain in the memory.

3. SuperPages and ARC

When adapting ARC for superpaging, some considerations should be taken into account. In this section, we would like to discuss these considerations and to see how they can affect the ARC algorithm.

3.1 Larger Pages

When using superpages the pages are occasionally larger. In such a case the recency is less important than the frequency. If a page is touched frequently, it can hint the operating systems that this page is important, even if the page is very large, while the importance of a touch of a larger page is less weighty.

3.2 Fragmentation

ARC and LRU do not take into their considerations the location of the “victim” that is chosen to be swapped out; hence they can leave many holes within a superpage, because some of the base pages that the superpage consist of can be in the memory while the others can be out of memory. When not all of the base pages are in the memory, a promotion can be costly. A better replacement algorithm must check the “neighborhood” of the victim page. A similar scheme was suggest by Romer et al. [10], but they preferred to use LRU.
3.3 Thrashing

Usually supperpages supported systems are less thrashing-proof than the old traditional paging systems. Obviously, when the memory is very large this deficiency is not critical. However, one of the importance advantages of the ARC over the “recency” algorithms is the thrashing-proof feature; hence the thrashing argument is clearly in favor of the use of ARC.

3.4 Coarse Granularity

The use of superpages causes all of the superpage’s base pages to be considered as important (or unimportant) ones, while just a few base pages are important or even just a single base page is important. In such cases the LRU gives bad results, because it has no mechanism to distinguish between the superpages that contains a lot of important base pages and the superpages that contains just a small number of important base pages. “Recency” algorithms cannot distinguish this difference; therefore LRU will not be a good choice. ARC, however, takes into account the “frequency”; thus ARC can be a better choice.

3.5 Gathering of the Accesses

When the page is small a look of the specific location of the readings/writings is not significant. However, one can think that a gather can imply an important page, while scattered readings/writings will mean arbitrary accesses. However, practically this assumption is not proved as correct.

4. Evaluation

ARC stipulates that the threshold of the “recency” list is $p$ which can be equal or less than the memory size. The outcome of the previous section is that this threshold is too high. The coarse granularity of the larger pages can be misleading and biased towards to the “recency” and against the “frequency”.

Reducing this threshold and put it on a lower value can do better. The results are changed from one benchmark to another and it depends on the memory size. It appears that in some cases the value can be reduced in order to give a better performance. Such a reduced threshold can be more reasonable threshold for superpages systems, because it
does not permit too much endowment in the “recency” list and enforce more weighty portion for the “frequency” list.

5. Conclusions

ARC gives quite remarkable results for many systems that have been used the traditionally LRU policy. In this paper we show that ARC is suitable for superpaging systems as well. Yet, an adjustment of the threshold can yield much better performance.

6. References


