CS 1550 – Introduction to Operating System (Summer 2016)

Project 3 – Virtual Memory Implementation

Mohammad Hasanzadeh Mofrad (hasanzadeh@cs.pitt.edu)

05/17/2016

---

1. Introduction

In this report we investigate the characteristics of 4 different page replacement algorithms – Optimal, Clock, Aging, and Least Recently Used (LRU) – based on a 32 bit implementation of a virtual memory in C (vimsim program). The page replacement algorithms are as follows:

Optimal: The baseline page replacement algorithm which knows about the future memory accesses. In essence, this algorithm is impossible in the real world applications. The purpose of bringing this is to provide a baseline for comparison.

Clock: The better implementation of the second chance algorithm which uses a referenced bit to identify the frequently used pages along with the First-In-First-Out (FIFO) strategy. The clock hand always points to the latest browsed page. It evicts pages using the FIFO policy but it will give a second chance to the pages which are referenced lately.

Aging: This algorithm is a variant of Not Recently Used (NRU) algorithm with a reference counter that shows the page recent uses by shifting and masking the reference bits. The reference bit has a limited memory which determined using a refresh parameter (reference counter length).

LRU: The LRU algorithm seeks to replace the page with the furthest access time. Thus, it stores the latest access time of each page for future page evictions.

In the following of this report, we conduct two experiments. The first one tries to determine the optimal value for the refresh parameter of the Aging algorithm and the second experiment compares different page replacement strategies.

2. Experiment 1: Finding the optimal value for Aging refresh parameter

In this experiment, we try to find the optimal value for refresh parameter of the Aging algorithm. In order to set this parameter, we use 8 pages and run the Aging algorithm on gcc.trace file. From Figure 1, one can see that **24** is the optimal length for the refresh parameter because this value is placed on the lower section of the plot (after the elbow) and has a slight page fault ratio difference (approximately 0.04) with the longest reference length.

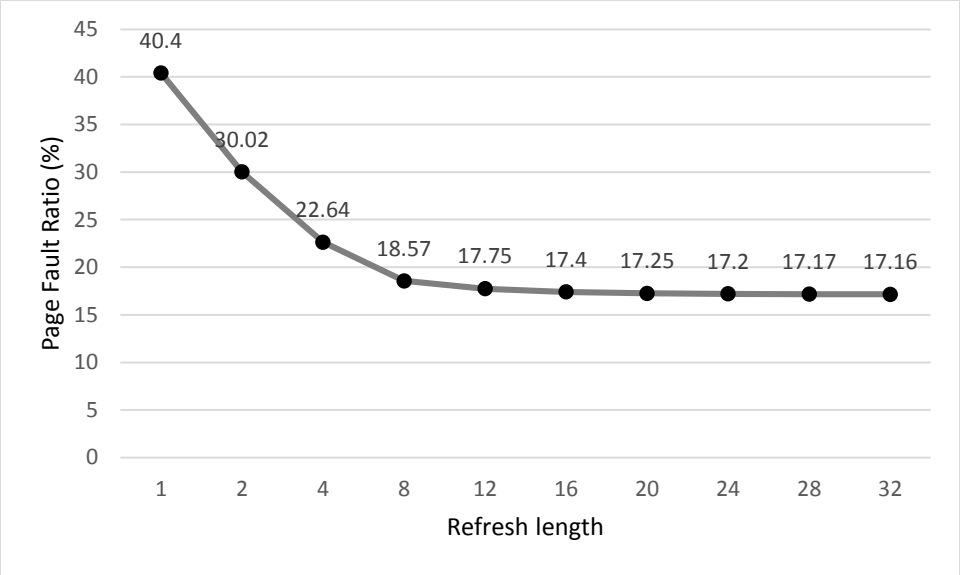

Figure 1. Page fault ratio for Aging algorithm with different number of frames.

3. Experiment 2: Comparing different page replacement algorithms

In this section we report the results for different page replacement algorithms using gcc and swim memory access traces. The experiments were conducted using 8, 16, 32, and 64 frames and the results are reported in Figure 2 and Figure 3. As you can see, the page fault pattern of these two figures is the same. So, the following discussion mainly tries to point the behavior of different algorithms on both of these figure.

From Figure 2 and 3, the Optimal page replacement algorithm has the best results whereas the Aging algorithm (with 24 as refresh window) has the worst performance. The Optimal algorithm can predict the future page requests and it can ultimately decide which one to replace. On the other hand, the Aging algorithm fails in both trace files. The Aging algorithm has a great start with 8 frames and performs the same as Clock and LRU but it cannot maintain its performance while increasing the number of frames. This observation is mainly due to the span of use feature of this algorithm (refresh parameter) combined with the current page reference that tends to fail the algorithm while having a page fault history that is not really correlated with the future use of pages.

The Clock and LRU algorithm tends to produce the same results in Figure 2 and Figure 3. The LRU algorithm evicts the page with the furthest access time, so it has a greedy policy that ignores the future dynamics of memory accesses. Still this short term history tends to achieve good results. The Clock algorithm which is the implementation of the second chance algorithm with a referenced bit will give a second chance to the frequently used pages while moving clock hands and evict a pages which its referenced bit does not set. Thus, the clock hand evicts the page that its referenced bit is not set. The implementation of this algorithm creates a circular list of pages on top of the frame list. Clock is a fast and efficient algorithm because the clock hand is always points

to the page to evict. Also, Clock algorithm exploits the latest information of the page usage so that it gains a better performance compare to LRU algorithm.

From Figure 2 and also Figure 3, while having only 8 frames, the Clock, Aging, and LRU algorithms have a same fault rate. Although, increasing the number of frames improves the performance of Clock and LRU, it degrades the performance of Aging algorithm. The Aging has a limited history of previous page faults (24 for this experiment). Since the access pattern of the trace files do follow a specific long term pattern, the Aging algorithm cannot exploit larger Frame numbers and tends to fail in this experiment while increasing the number of frames.

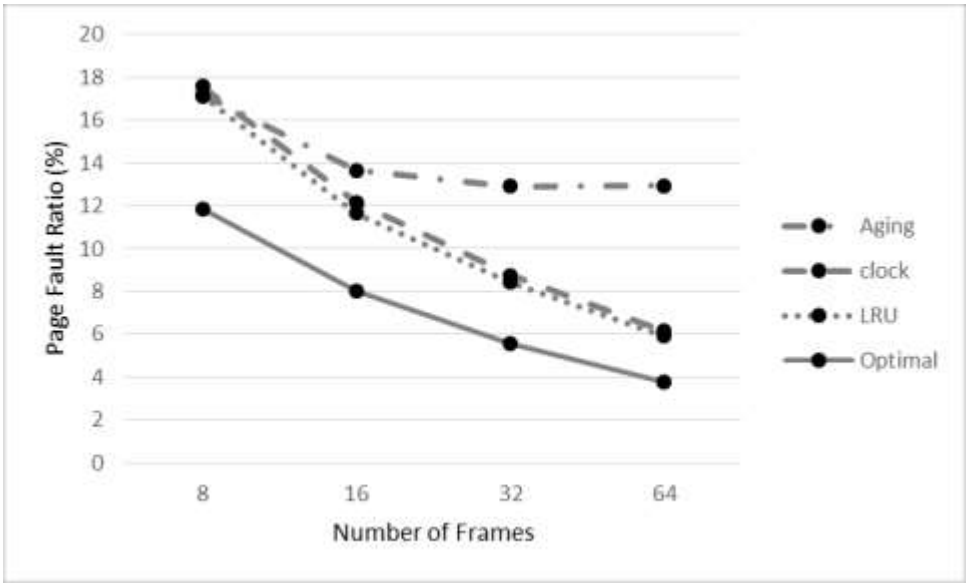

Figure 2. Comparison of different page replacement algorithms on gcc trace.

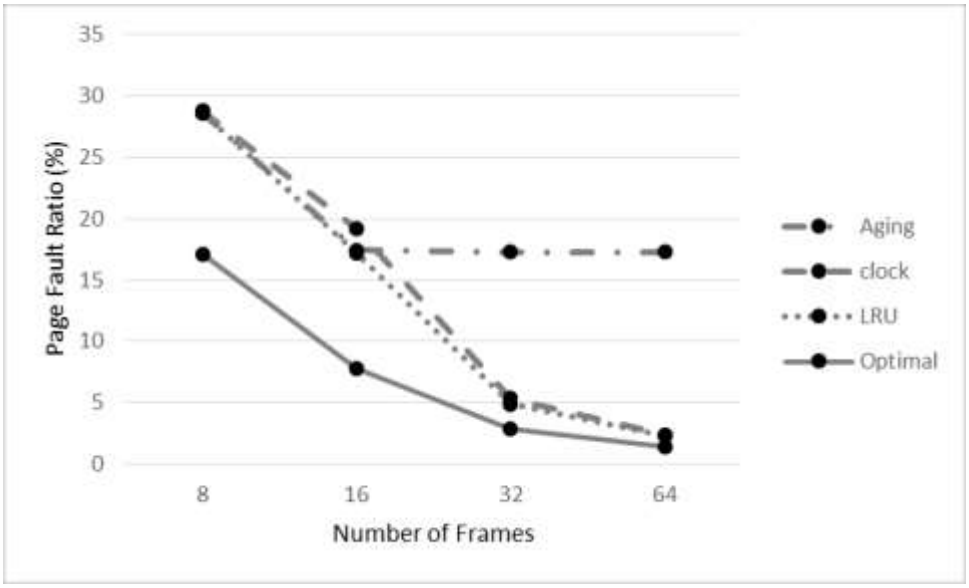

Figure 2. Comparison of different page replacement algorithms on swim trace.

4. Conclusion

In this report, we investigate the behavior of Optimal, Clock, Aging and LRU page replacement algorithms. One of the interesting observations from Figure 2 and Figure 3 is that except for Aging algorithm by increasing the number of frames the page fault rates decreases. The Optimal algorithm has the best results but it is not a practical solution. Next, the Clock and LRU algorithms perform similarly except for a slight performance improvement for Clock algorithm. Finally, the Aging attains the worst results.