Drew Kozak

Dr. Misurda

CS1550 Operating Systems

15 July 2018

Project 3 Writeup:

The purpose of this project was to look into four different Paging replacement algorithms, Optimal, FIFO, Clock, and NRU. I generated comparisons and attached graphs showing page faults and the disk writes associated with the number of frames increasing to represent physical memory. We were given three different files, swim.trace, bzip.trace, gcc.trace so output differed per each trace file given. I am attaching another file with more output sample called ‘project 3 alg recordings’ to give a better reference for all of these different algorithms being tested.

FIFO in general is a pretty bad paging replacement algorithm, Dr. Misurda even said so in class. FIFO is going to keep track of the order pages were brought into memory and keep them in a linked list or queue. Removing the oldest page becomes a little irrelevant at this point, but that page could still be in use, so you are now evicting a page that has a real purpose. FIFO is usually cheap and intuitive, but if using a linked list, relinking all the pages in memory causes a lot of overhead as well. FIFO also encounters Belady’s anomaly, that states as the number of page frames increases, there is also an increase in the number of page faults for the algorithm. The larger the memory being used, the higher the amount of page faults involved, usually doubling in page faults.

Optimal is a base, something we can never really implement due to not actually knowing the future. That being said, if you plan on working with a ton of memory accesses being performed on pages where the majority are dirty already, Clock algorithm is most likely the best. That is because clock gives priority to the pages that are consistently referenced in between page evictions, in comparison an algorithm like NRU might evict a page that was used only fifteen memory accesses ago because of the reference bit being refreshed. Going on top of that, the refresh rate is when NRU resets the referenced bit of all frames in RAM after N instructions. Trying to optimize NRU means finding a medium where there is the least amount of disk writes and the least amount of page faults involved. There will be a lot less page faults at higher refresh rate like 75, while there will be a lot less disk writes at let’s say 25, sometimes a happy medium must be taken that makes a compromise, let’s say where N = 50.

All things considered, since Optimal is not truly achievable, NRU has some hiccups, and we knew from the very start FIFO would not be very good, Clock seems to be the winning page replacement algorithm for implementing into an OS. Clock has very good ratios in regards to page faults and writes to disk when the number of frames are sixteen, thirty two, or sixty-four compared to the other algorithms. Clock is also a quite practical implementation to use in an operating system.