Dan Salerno and Jaymes Garland

CS 492

HW 2

We pledge our honor that we have abided by the Stevens Honor System

What we expected to see:

We expected FIFO to be the worst because it is the simplest algorithm. Since the page in memory the longest may be often replaced, this should have a higher number of page replacements. Clock should be the 2nd worst because it’s just an improved FIFO, so it should perform better. LRU should be the best because it uses the locality property (pages used recently will be used again soon), so pages that are used frequently will stay in memory, lowering page replacements. Pre-paging should decrease page replacements by keeping pages that are close in memory loaded, since if you are accessing one page, you will likely be accessing the next contiguous pages often. Lastly, the number of page replacements should go down as page size increases, since each page holds more memory and is more likely to hold a memory location that you need.

What we saw:

FIFO was the worst, followed by Clock and then LRU, like we expected. However, FIFO and Clock actually increased page replacements when page size was increased to 16. We think that is the case because with page size of 16 and the provided plist file, each process only gets 3 pages, so there is a smaller chance that the memory location you are trying to access is held on one of those 3 pages. For pre-paging, all algorithms had less page replacements than they did during demand paging, which is what we expected. However, FIFO and Clock increased in page replacements various times. We think this occurred because as page size grows, less pages are held in memory per process, and since prepaging removes two pages every time, there is a higher chance that a page we will need will be removed, causing more page replacements in the future.

Complexity:

All three algorithms were relatively simple to implement, however LRU was the easiest because you only needed to find the minimum of one page value to find what page needed to be replaced. FIFO and Clock had similar complexity since they both needed an additional data structure (queue and circular buffer, respectively). Clock also had the extra overhead of keeping track of the R (modified) bit for each page.

Randomness changes:

With a random access trace, all algorithms would be much worse. LRU depends on the locality principle which doesn’t exist with random traces. Clock tries to keep in frequently used pages, which is pointless if it’s all random. FIFO would not be too much worse, but the probably of a page being in the FIFO queue when it is selected randomly would be less, causing more page swaps.