

Project 5: Virtual Memory

By Andrew Paek (apaek1) and Douglas Schmieder (Dschmied)

This assignment is to implement a paged virtual memory and learn how to implement an OS fault handler. We set up this experiment by creating virtual memory and a frame table that represents physical memory. We took the elements in the virtual memory, wrote them to disk and updated the physical memory to hold these elements. Once the physical memory was full, we had to implement a page fault handler that chose a frame in the table to replace. We ran these tests on student machine 00, and the command line arguments are `“./virtmem npages nframes rand|fifo|custom scan|sort|focus.”` To compile the project, you just have to run the make file using the command `“make”`.

The custom page fault handler algorithm utilizes pseudo-randomness to choose the frame to replace. It first checks for any dirty bits (bits with both `PROT_READ` and `PROT_WRITE` set) and uses these frames first. If they do not exist, it just re-implements the `RAND` algorithm, picking a random page to replace. After picking a random page, we make sure that this page is inside the frame, and then replace it with the new page.

In the following diagrams, we have tested the three page fault handler algorithms with 7 different frame values (3, 10, 20, 40, 60, 80, 100) to see the trend of each algorithm. As we increase the frame number, it is obvious that all three algorithms improve significantly. Interestingly enough, there are several instances where `RAND` outperforms `CUSTOM` and other instances where `CUSTOM` outperforms `RAND`. This is most likely due to RNG, where `RAND` may get a lucky set of frames to replace, which improves its performance. However, because it is random, overall we can see that `CUSTOM` is equally as good as `RAND` (in the case of `Scan`), or significantly better (in the case of `Sort`). `CUSTOM` also outperforms `FIFO` in `Scan`, indicating that `Scan` is an inefficient program for `FIFO`. In any other case where `CUSTOM` outperforms `FIFO`, it is most likely due to the randomness of the frames chosen to be replaced.

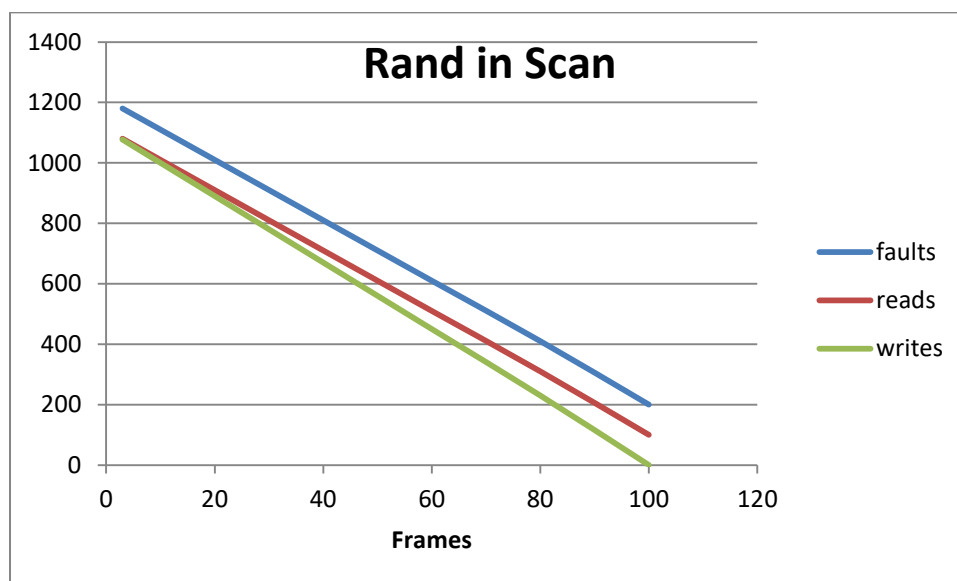


Figure 1: Rand in Scan

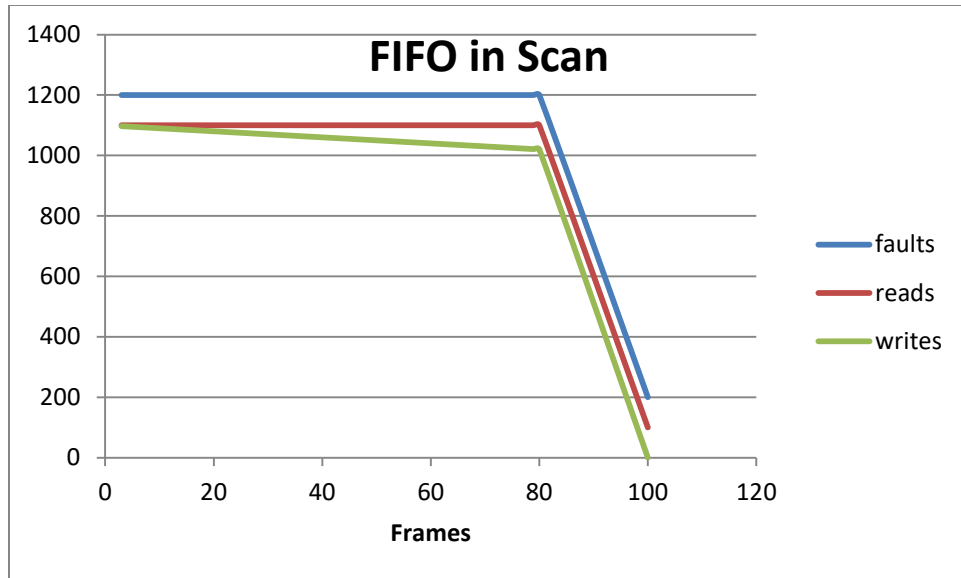


Figure 2: FIFO in Scan

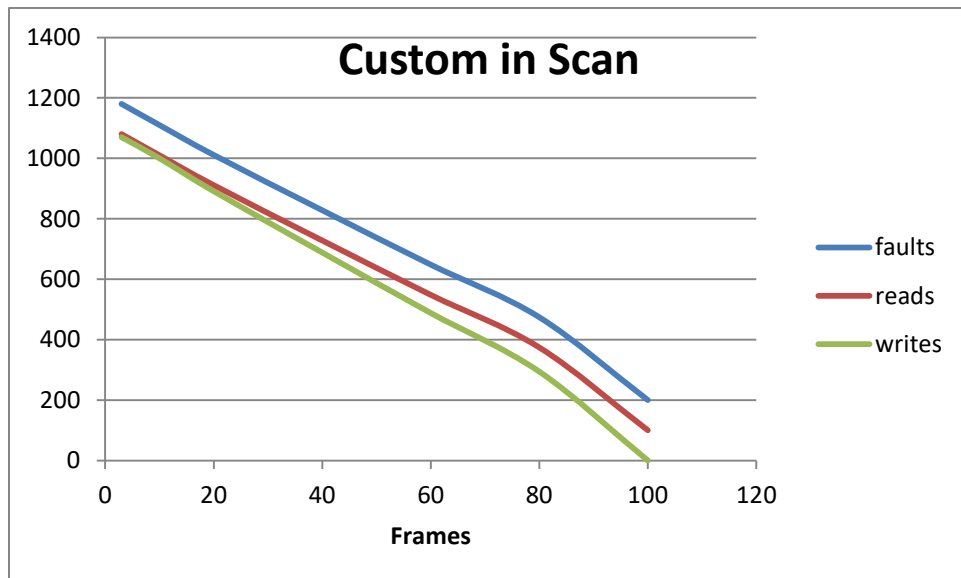


Figure 3: Custom in Scan

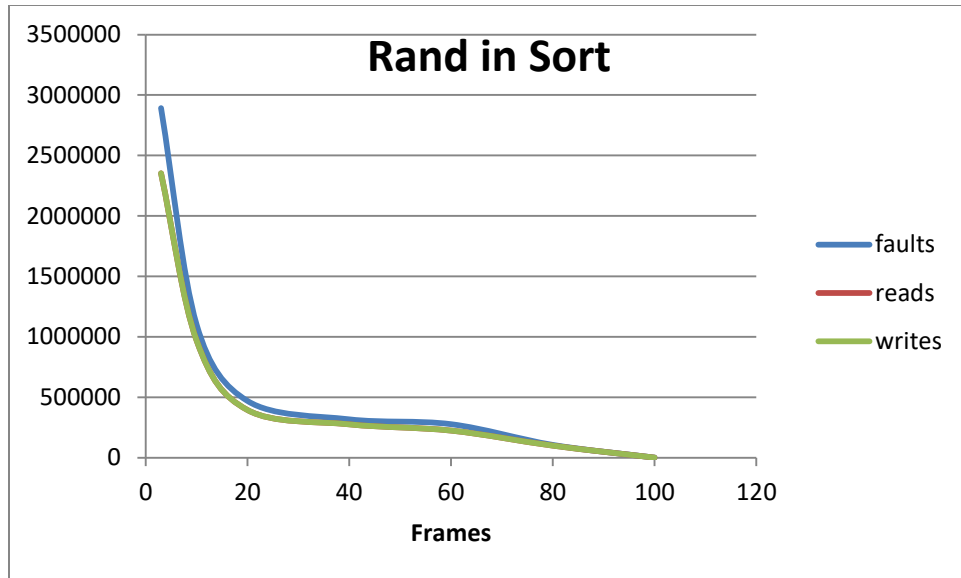


Figure 4: Rand in Sort

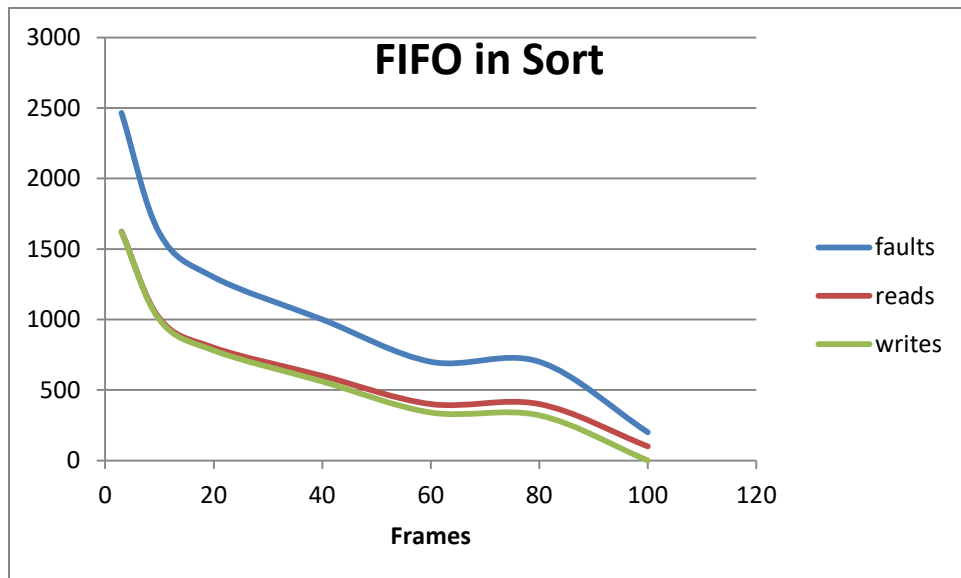


Figure 5: FIFO in Sort

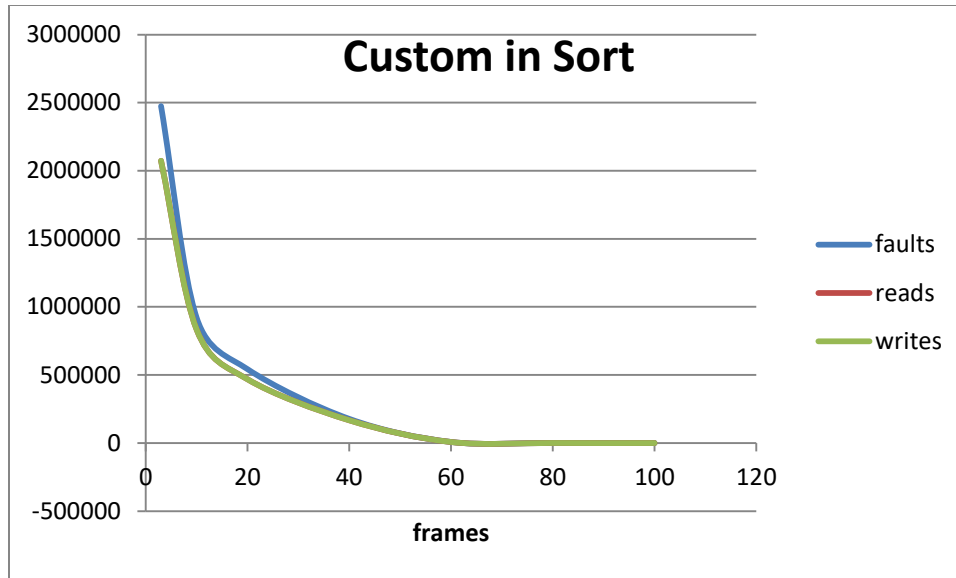


Figure 6: Custom in Sort

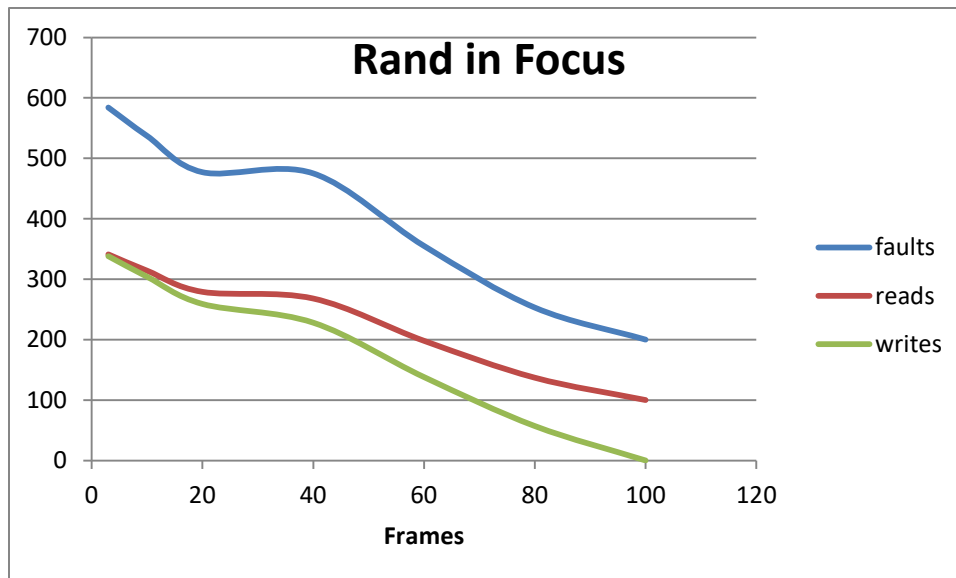


Figure 7: Rand in Focus

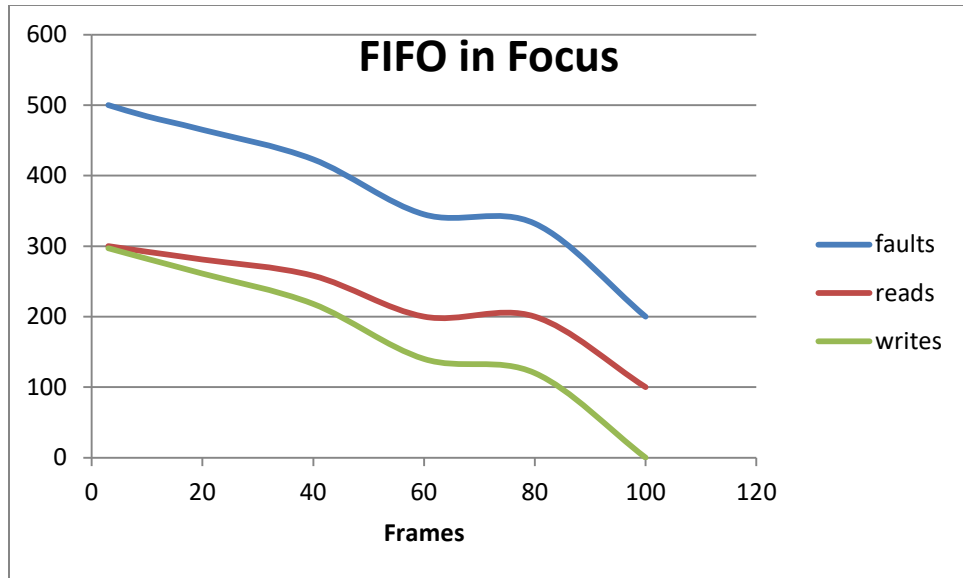


Figure 8: FIFO in Focus

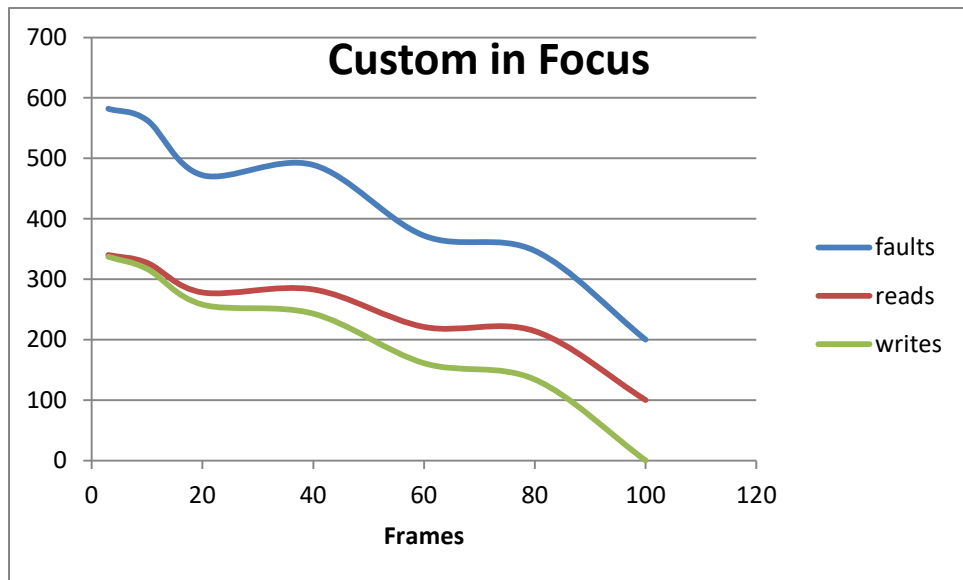


Figure 9: Custom in Focus