Virtual Memory - Page Fault Handler Experiment

Samuel Thorson, Luke Lopata

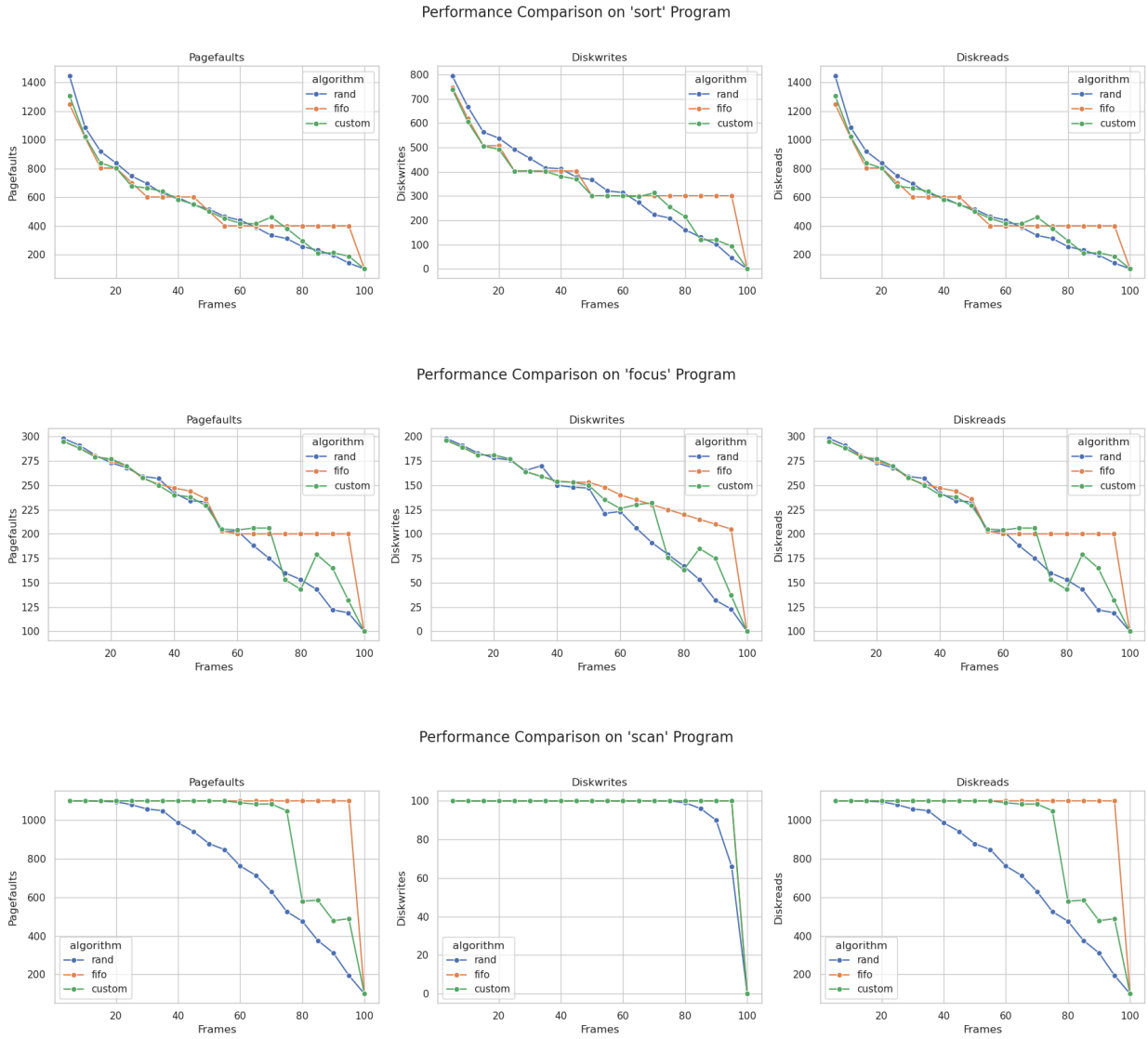
Experiment purpose and setup:

The purpose of these experiments is to compare the performance between 3 page fault handler algorithms: random, fifo, and our own custom algorithm. We ran these experiments on a local linux machine. To reproduce our experiment you should run the command: `./test.sh`. This runs all 3 algorithms against 3 different programs: sort, scan, focus. Each algorithm is run multiple times for each program in order to show how the algorithms perform for different ratios of pages and frames. The output shows the total number of page faults, disk reads, and disk writes for each algorithm / program pair. Running this command will populate the data and graphs in the `outputs` subdirectory.

Description of custom page handler:

The custom page fault handler we created is meant to emulate the ‘clock’ page fault handler. The handler attempts to prevent pages that have been recently used from being evicted. All pages that have been assigned a frame are tracked in a list: clock_entries. Each of these entries contains a boolean meant to represent a use bit. When the program page faults and adds the write bit to a page, the use bit for that clock entry gets set to true. To evict a page, the algorithm moves around the clock, looking for an entry that does not have the use bit set to true. As it moves along, if it encounters a use bit set to true, it flips the bit back to false. This approximates an algorithm that finds and evicts the page that wrote to the disk longest ago. This is important because writing to the disk requires a significant amount of overhead, so minimizing the number of disk writes should improve the performance.

Results:


Performance Comparison on 'sort' Program
Pagefaults
Diskwrites
Diskreads
Frames
algorithm
rand
fifo
custom
Performance Comparison on 'focus' Program
Pagefaults
Diskwrites
Diskreads
Frames
algorithm
rand
fifo
custom
Performance Comparison on 'scan' Program
Pagefaults
Diskwrites
Diskreads
Frames
algorithm
rand
fifo
custom

Analysis:

As expected, for each algorithm, as we increase the number of frames, the number of page faults, disk reads, and disk writes decreases. However, what is most surprising is that FIFO is the worst performing algorithm out of the three. We hypothesize that this is due to the fact that FIFO does not prioritize heavily used pages, instead it bases its evictions on how long ago each page was assigned a frame. This could lead to a scenario where a page that is repeatedly used still gets evicted just as often as a page that is used infrequently, causing the plateaus you can see in the graphs above. As for random and our custom algorithm, random performs much better than custom on the scan program. This is due to the fact that scan doesn't rely on many disk writes, so the optimizations we made to limit evicting pages with the write bit set doesn't play a major role in performance. As stated in the explanation of the custom algorithm, the use bit for a page is only set on disk writes, not disk reads. On sort and focus, random and custom perform about the same. While we expected our custom algorithm to perform much better than random, it is possible that the resources saved in limiting evictions for recently used pages did not make enough of a difference to show a significant performance improvement in the graphs.