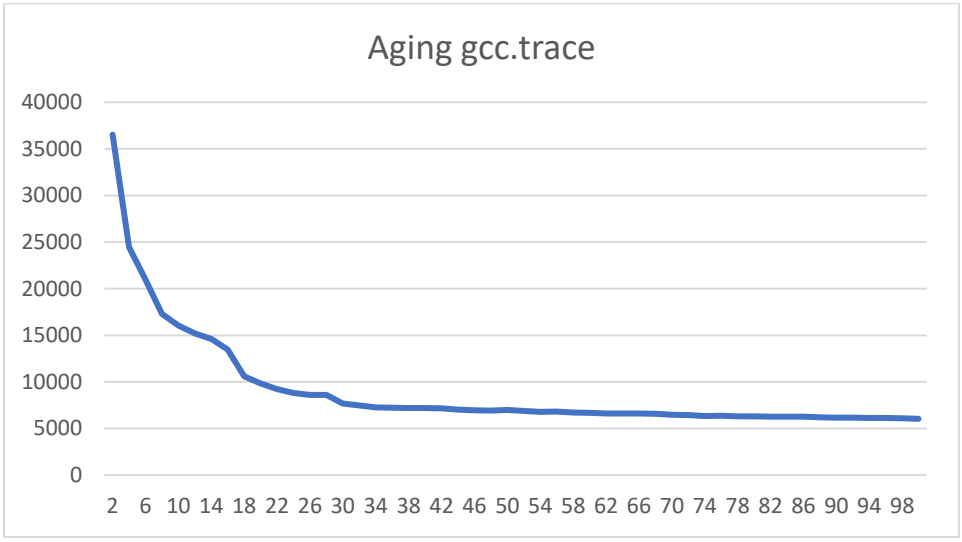
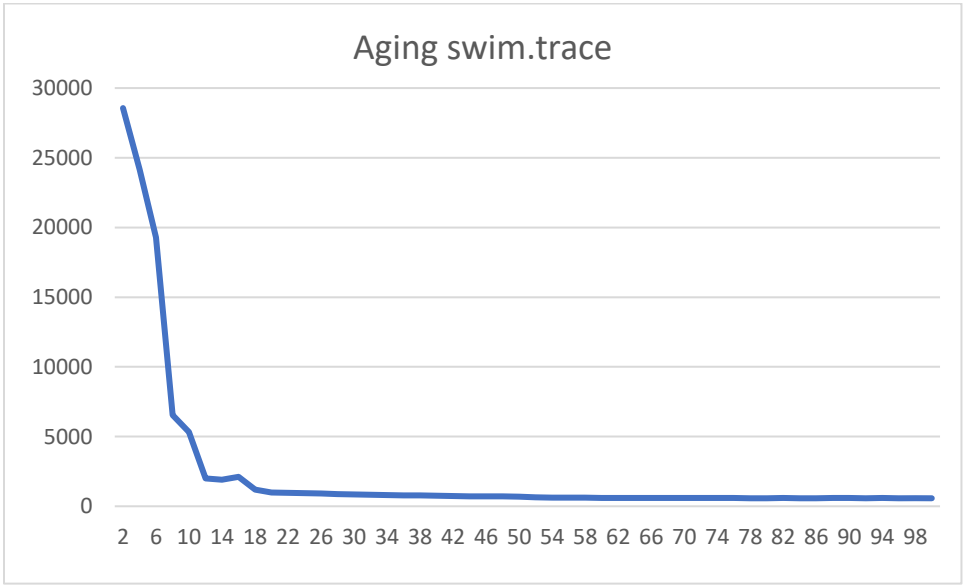
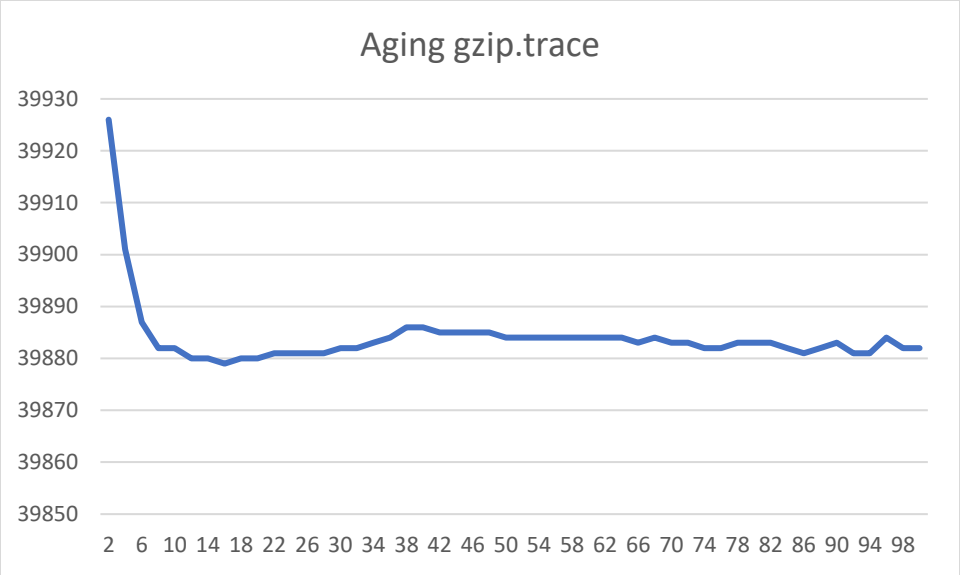
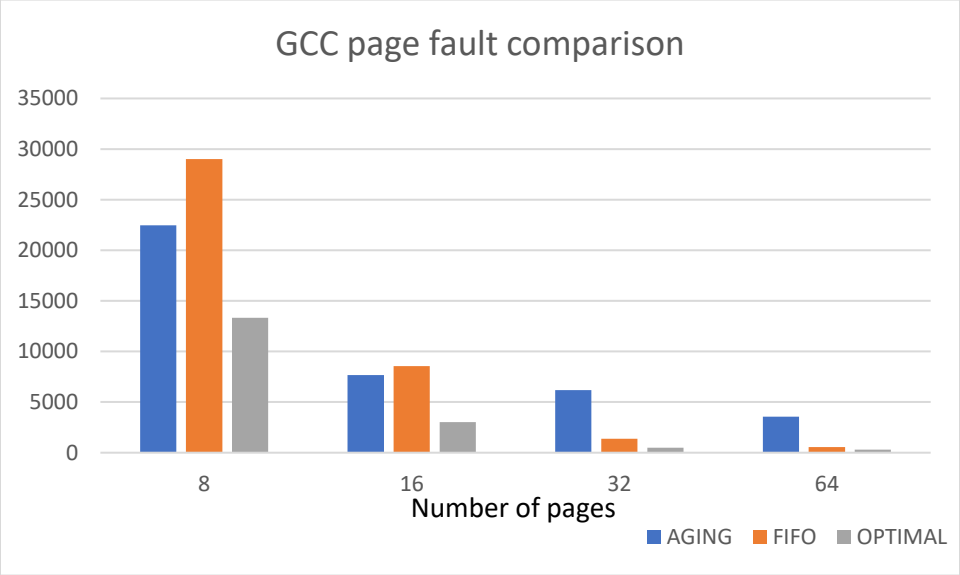
To try to find a good refresh period, I decided to run each of the trace files (gcc.trace, gzip.trace, and swim.trace) on my aging algorithm using 16 page frames. I started by having the refresh rate at 2 and incremented the refresh rate by 2 on each iteration all the way up to 100. I output the number of page faults for each file per refresh rate to a text file, and then placed the results into graphs. Based off of the graphs, aging with gcc.trace really started to decrease its variation in the number of page faults around a refresh rate of 30. Aging.trace started to decrease its variation in the number of page faults at a refresh rate of around 18. Lastly, gzip.trace started to decrease its variation in the number of page faults at a refresh rate of around 6. After looking at the graphs, it did not seem as if it made much of a difference in the number of page faults by increasing the refresh rates after the stated rates above for the 3 files. The rates page fault rate continued to go down, but by much smaller margins (less than a few hundred). I decided a refresh of 30 would be best just because gcc.trace really did not start to level out until this point. I also used 32 page frames to compare my results to but it really did not make that big of a difference. The number of page faults were lower with 32 page frames, but the graphs still came out looking similar.

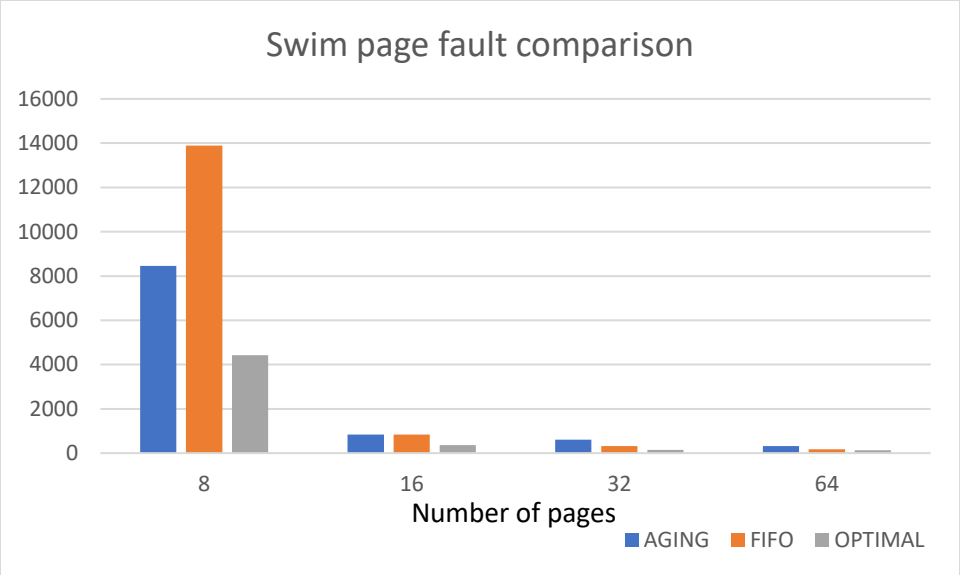


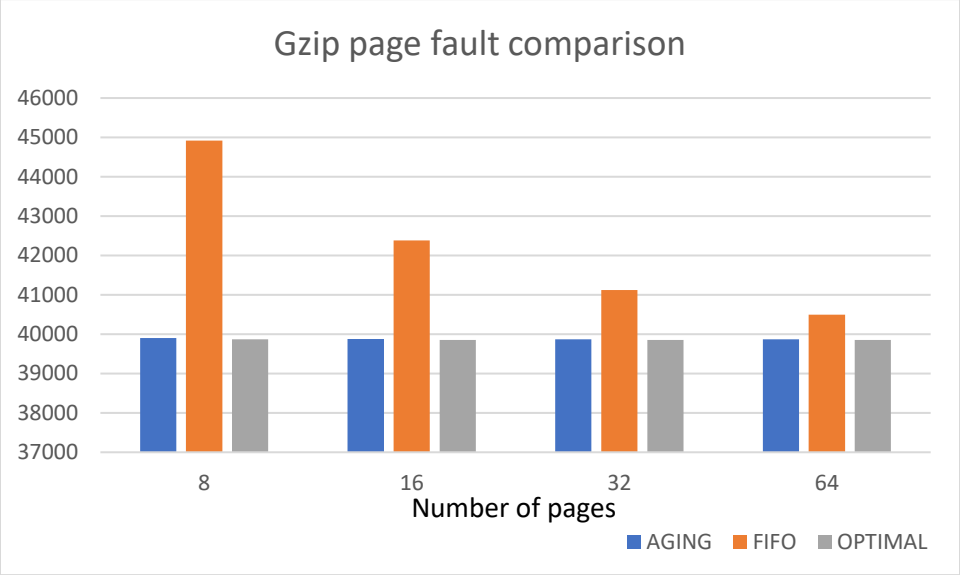




I then ran each of the three algorithms with 8, 16, 32, and 64 frames to compare page fault statistics. I also placed these into graphs based off of the trace file to make the results easier to understand. For gcc.trace, aging had much closer results to optimal than FIFO for 8 and 16 frames but was surprisingly worse than FIFO for 32 and 64 frames. For swim.trace, aging again had much closer results to optimal than FIFO for 8 and 16 frames, but was then again worse than FIFO for 32 and 64 frames. Lastly, for gzip.trace aging was a better algorithm for 8, 16, 32, and 64 page frames than FIFO was. Based off of these results, I think that aging is better in most circumstances, if you are strictly looking at the number of page faults that it causes. Aging is drastically better for a low number of frames, such as 8. As the number of pages increase, FIFO seems to drastically improve. With 64 frames, FIFO was either better than aging or just barely worse than it, leaving me with the conclusion that aging is better when the system has few frames, and FIFO is better when the system has more frames.







In order to find any instances of Belady's anomaly in FIFO, I made my program run the FIFO algorithm on 2 to 100 page frames for each file. I output the results into a separate text file for each trace file, which simply held the number of page faults caused by the number of page frames. I then made another program to read through these text files, and mark whenever the next number was bigger than the previous. I found that the only instance of Belady's anomaly between the three files was with gcc.trace. With 91 page frames, there were 457 page faults, but for 92 and 93 page frames, there were 458 page faults.