# Emily Hua
# CS1550: Project 3 Write Up

## FIFO Analysis

Belady's anomaly occurs when we implement the First In First Out (FIFO) page replacement algorithm. This anomaly is the phenomenon of when memory size increases, the page faults increase as well. This is very odd behavior because supplying more memory shouldn't result in worse performance. In Belady's anomaly, more memory leads to more page faults. I've tested this out with my implementation of FIFO for project 3. Below is a table that includes some data I've collected while running the implementation against the gzip, swim, and gcc files. The boxes within are the number of page faults in correspondence to the file and the number of frames. As you can see, there are some instances of Belady's anomaly. For example, from 46 frames to 55 frames, in the gzip file, the number of page faults increased from 790 to 40598. In the gcc file as well from 46 frames to 55 frames, the number of page faults increased from 384 to 651. However, those appear to be the only instances of Belady's anomaly. After all, it is an anomaly and may not happen too often. Also, after those anomalies, it is interesting to see that it goes back to normal, where as the number of frames increases, the number of page faults decreases. It seems that Belady's anomaly acts as a hiccup, due to its random, peculiar, and rare occurrence.

| Frames | 2 | 8 | 32 | 46 | 55 | 64 | 77 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|
| Gzip | 60446 | 44918 | 41120 | 790 | 40598 | 40496 | 40391 | 40316 | 40272 |
| Swim | 81595 | 13893 | 326 | 226 | 192 | 177 | 154 | 148 | 147 |
| Gcc | 127688 | 29011 | 1375 | 384 | 651 | 511 | 491 | 460 | 436 |

*Values inside the boxes are the number of page faults.

## Statistics: Page Faults v Frames Explanations

I've included some tables below that report the page fault and disk write statistics for 8, 16, 32, and 64 frames. I've also included some graphs that plot the number of page faults versus the number of frames. The optimal page replacement algorithm seems to have the best results in terms of least amount of page faults. As you can see from the tables below, throughout all of the number of frames, the page faults for the three trace files are very low compared to FIFO and aging. The number of disk writes are also very low for optimal. However, optimal is not exactly always possible in a real operating system because it is hard to see the future. Thus, the algorithm that may be most appropriate for use in an actual operating system may be the first in first out page replacement algorithm, mostly because it is the most possible to implement. The

aging algorithm also seems to perform the worst of the three. Though, my results for aging may be a little off because my aging algorithm produces results that are a couple of hundred page faults off. However, if I were to make a conclusion based on the data that I have, I would first say optimal has the best performance in terms of lowest page faults and disk writes, and would be preferable in an actual operating system. Though, if we were not able to see the future, then FIFO would be the preferred algorithm in an actual operating system.

*FIFO*

| Frames | 8 | 16 | 32 | 64 |
|---|---|---|---|---|
| Gzip | 44918<br>39894 | 42384<br>39856 | 41120<br>39825 | 40496<br>39793 |
| Swim | 13893<br>8499 | 844<br>470 | 326<br>158 | 177<br>69 |
| Gcc | 29011<br>11519 | 8568<br>3542 | 1375<br>660 | 511<br>258 |

*Values inside the boxes above the line are the number of page faults, below the line are disk writes.

*OPT*

| Frames | 8 | 16 | 32 | 64 |
|---|---|---|---|---|
| Gzip | 39874<br>39845 | 39856<br>39830 | 39856<br>39830 | 39856<br>39830 |
| Swim | 4417<br>2183 | 358<br>152 | 183<br>85 | 183<br>85 |
| Gcc | 13328<br>4381 | 3020<br>1020 | 523<br>243 | 444<br>205 |

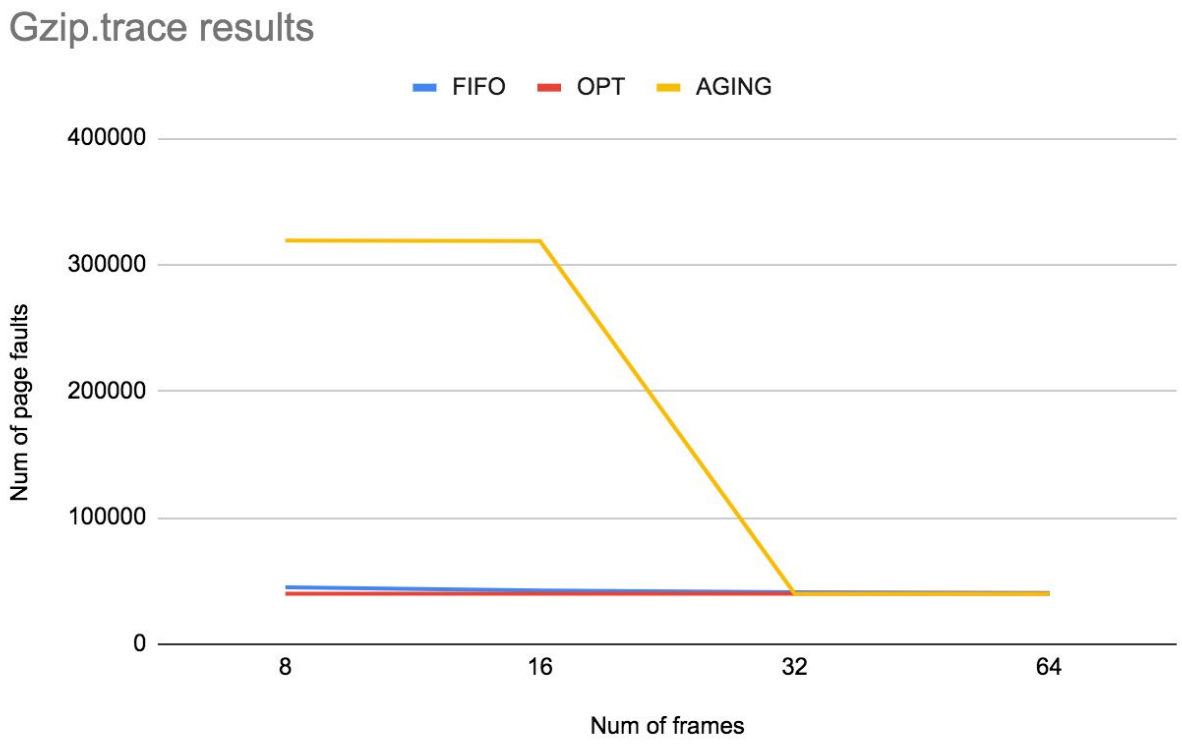
*Values inside the boxes above the line are the number of page faults, below the line are disk writes.

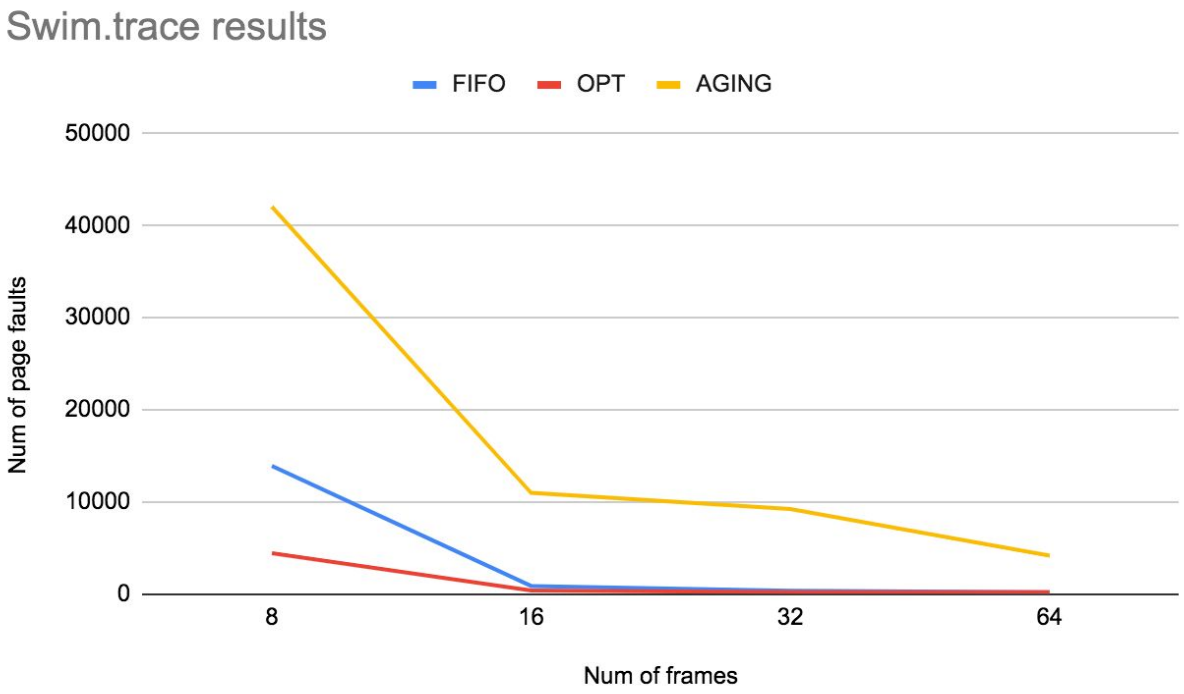
*AGING*

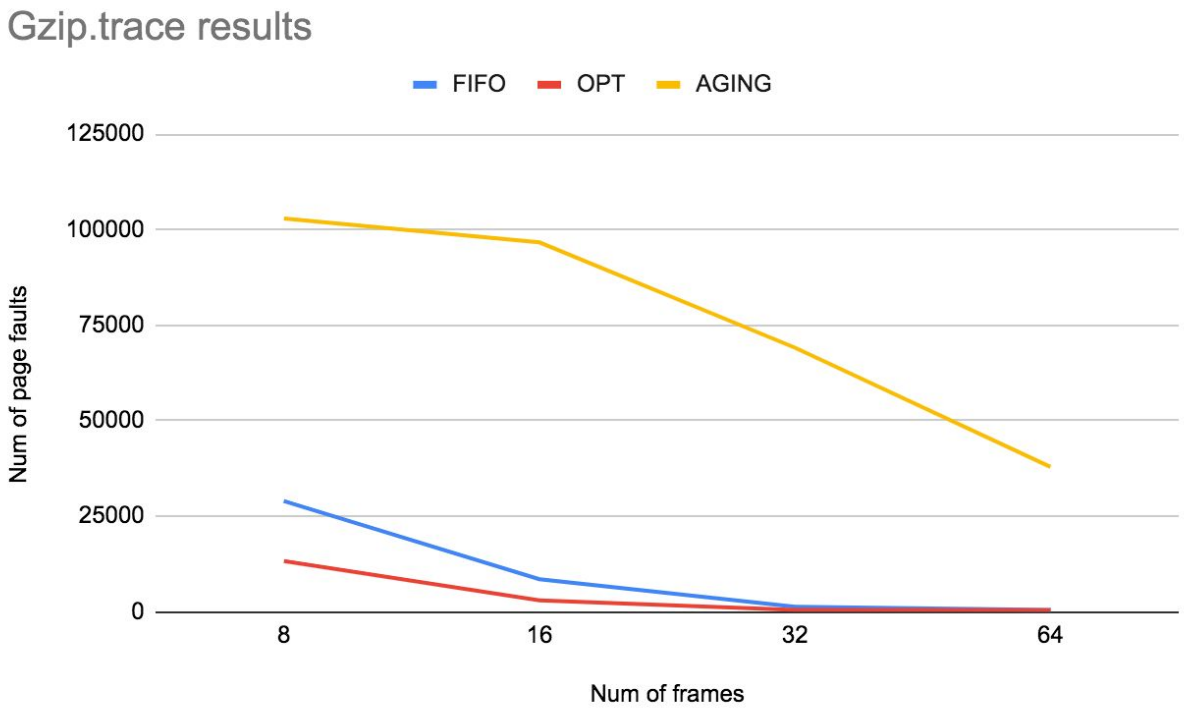
| Frames | 8 | 16 | 32 | 64 |
|---|---|---|---|---|
| Gzip | 319024<br>159494 | 318684<br>159345 | 39863<br>39830 | 39863<br>39798 |
| Swim | 42052<br>16659 | 10963<br>6118 | 9211<br>4874 | 4139<br>1706 |
| Gcc | 102916<br>28613 | 96694<br>26209 | 69143<br>22162 | 37941<br>9663 |

*Values inside the boxes above the line are the number of page faults, below the line are disk writes.

*Refresh rate of 5






## Optimal Value of the Refresh Parameter

Below is a table and graph that plots the page faults and disk writes for a range of refresh rates. My aging algorithm must not have been implemented correctly, as the number of page faults and disk writes stay consistent no matter what the refresh rate is. Thus, it will be difficult for me to come to a conclusion as to which is the most optimal refresh rate. I also have not included a graph as it will be redundant, since the data is consistent. However, I will draw up a conclusion based on what I would have expected to happen. We would like to find a refresh rate that has a good compromise between the least amount of page faults and disk writes. I believe it should also depend on the number of frames. If we have a larger number of frames, we would need a refresh rate that can accompany that size of memory. Therefore, the optimal refresh rate would depend on the number of frames and how much memory is allocated.

*Aging, num of frames: 16*

| Refresh Rate | 5 | 25 | 50 | 100 |
|---|---|---|---|---|
| Gzip | 318684<br>159345 | 318684<br>159345 | 318684<br>159345 | 318684<br>159345 |
| Swim | 10963<br>6118 | 10963<br>6118 | 10963<br>6118 | 10963<br>6118 |
| Gcc | 96694<br>26209 | 96694<br>26209 | 96694<br>26209 | 96694<br>26209 |

*Values inside the boxes above the line are the number of page faults, below are the disk writes.