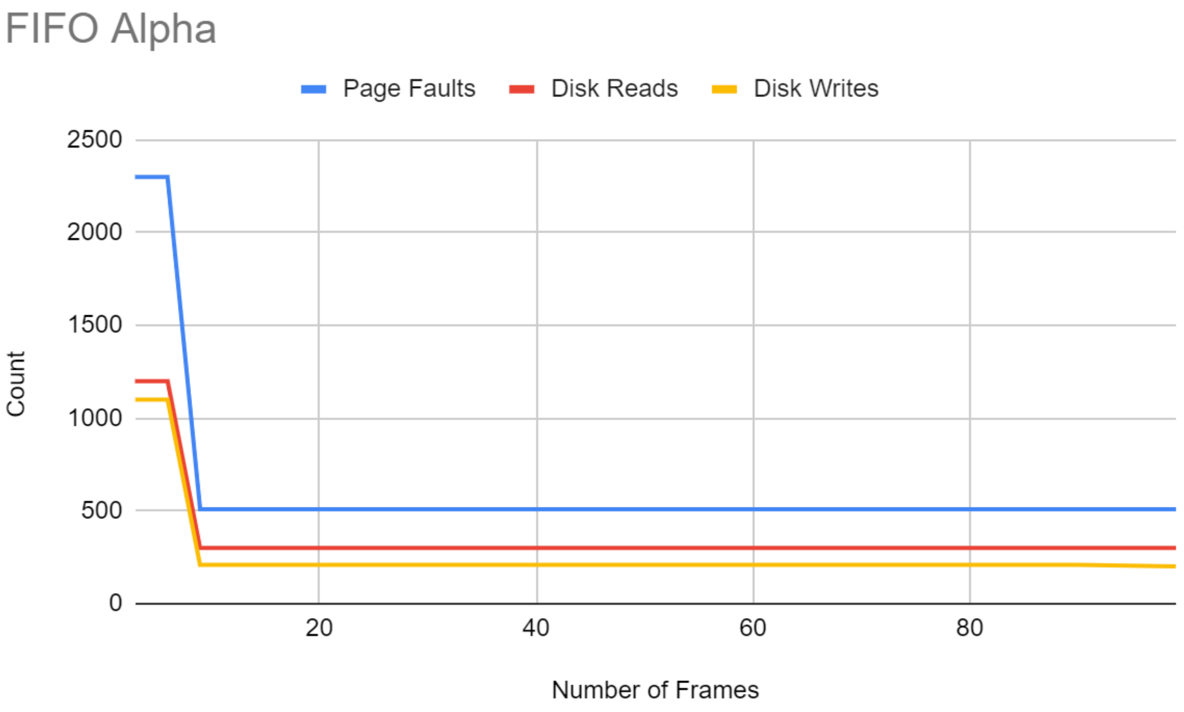
In your own words, briefly explain the purpose of the experiments and the experimental setup. Be sure to clearly state on which machine you ran the experiments, and exactly what your command line arguments were, so that we can reproduce your work on the same machine in case of any confusion.

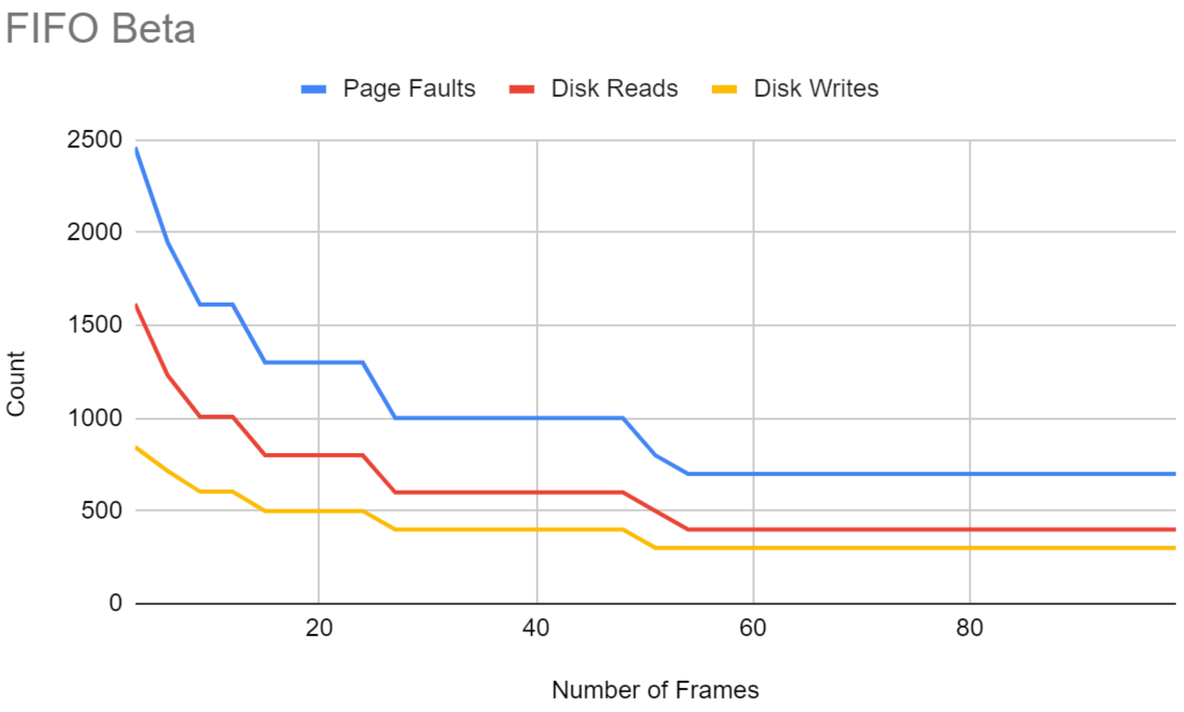
The purpose of this experiment is to compare the efficiency in memory management of different page replacement algorithms. We are using the total number of page faults, disk reads, and disk writes as performance metrics. I ran the experiments on student11, using all possible combinations of algorithms (rand, fifo, and custom) and programs (alpha, beta, gamma, delta). As per the instructions, the number of pages was kept constant at 100, while the number of frames ranged from 3 to 99, in steps of 3.

Very carefully describe the custom page replacement algorithm that you have invented. Make sure to give enough detail that someone else could reproduce your algorithm, even without your code.

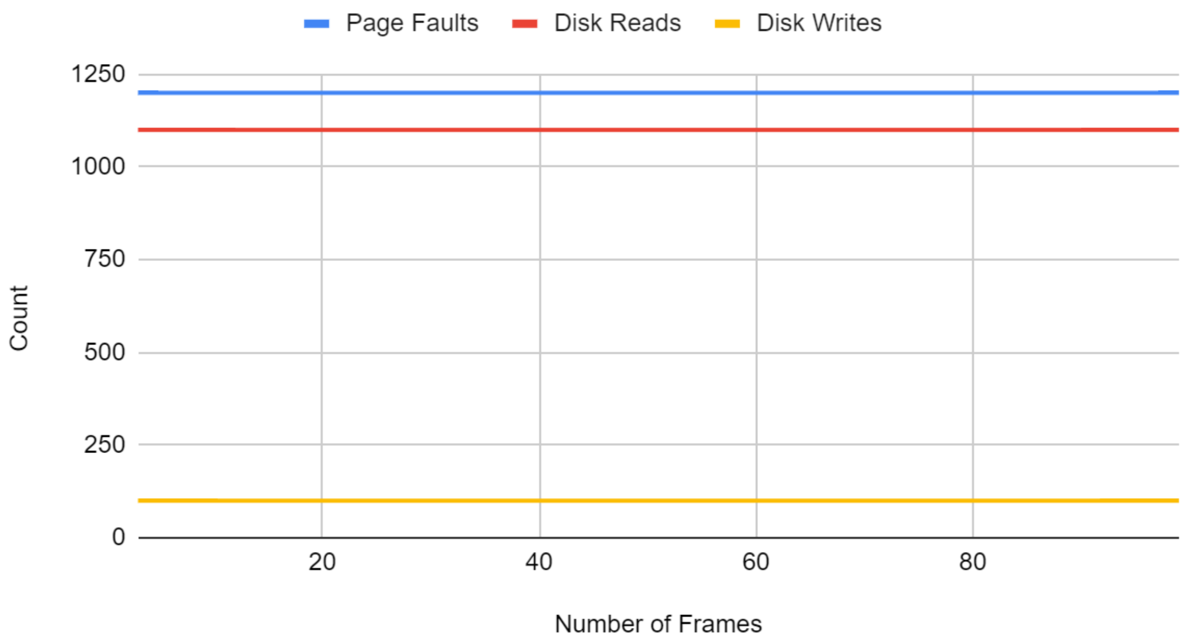
In my custom page replacement algorithm, I evicted 2 pages instead of just one. I always evicted the page in frame 0, and loaded the new page into that frame. Additionally, I evicted the final page in the frame table, and marked it as available.

Measure and graph the number of page faults, disk reads, and disk writes for each program and each page replacement algorithm using 100 pages for the application, versus NPF (the number of page frames), for NPF from 3 to 99 in steps of 3. Spend some time to make sure that your graphs are nicely laid out, correctly labeled, and easy to read. Please try to combine plots of the same *dependent* variable (e.g., number of page faults) for different algorithms. Don't try to plot different dependent variables on the same plot.

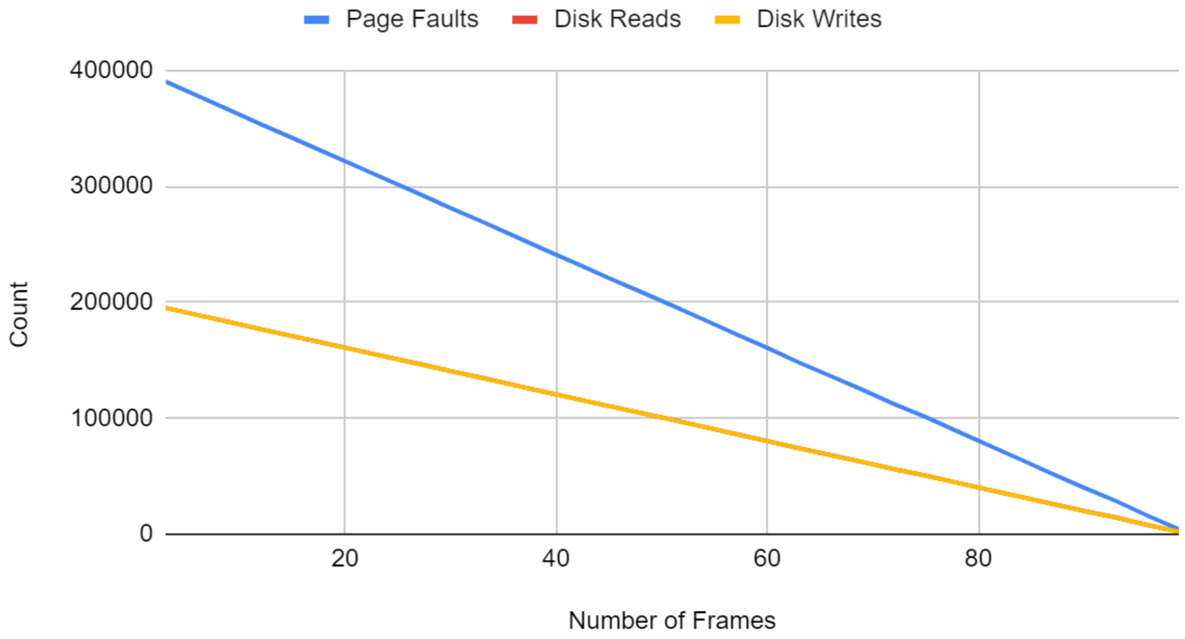

FIFO Alpha
Page Faults
Disk Reads
Disk Writes
2500
2000
1500
1000
500
0
Count
20
40
60
80
Number of Frames

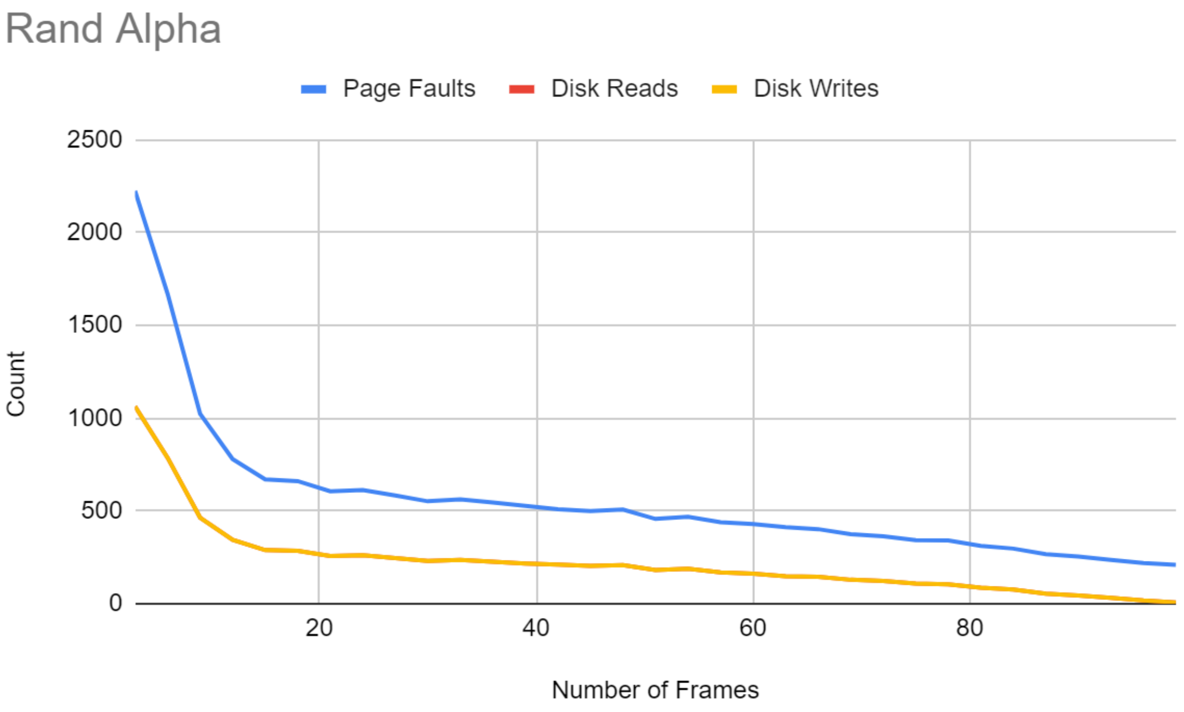

FIFO Beta
Page Faults
Disk Reads
Disk Writes
2500
2000
1500
1000
500
0
Count
20
40
60
80
Number of Frames

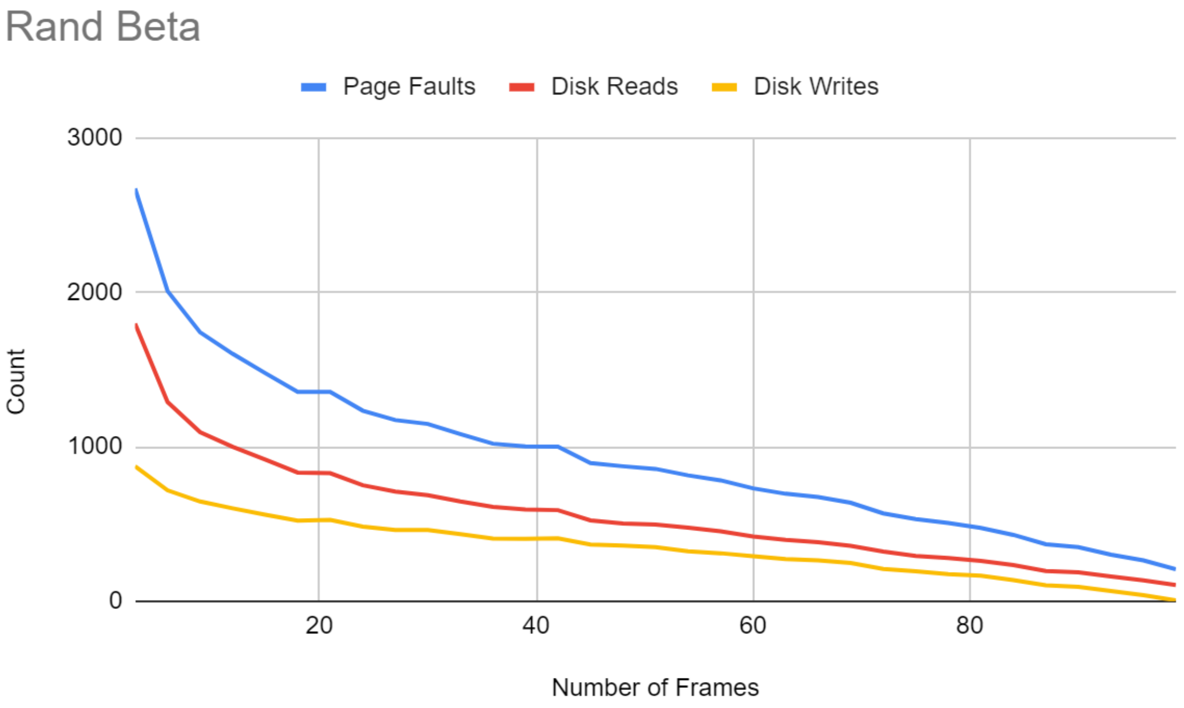
FIFO Gamma

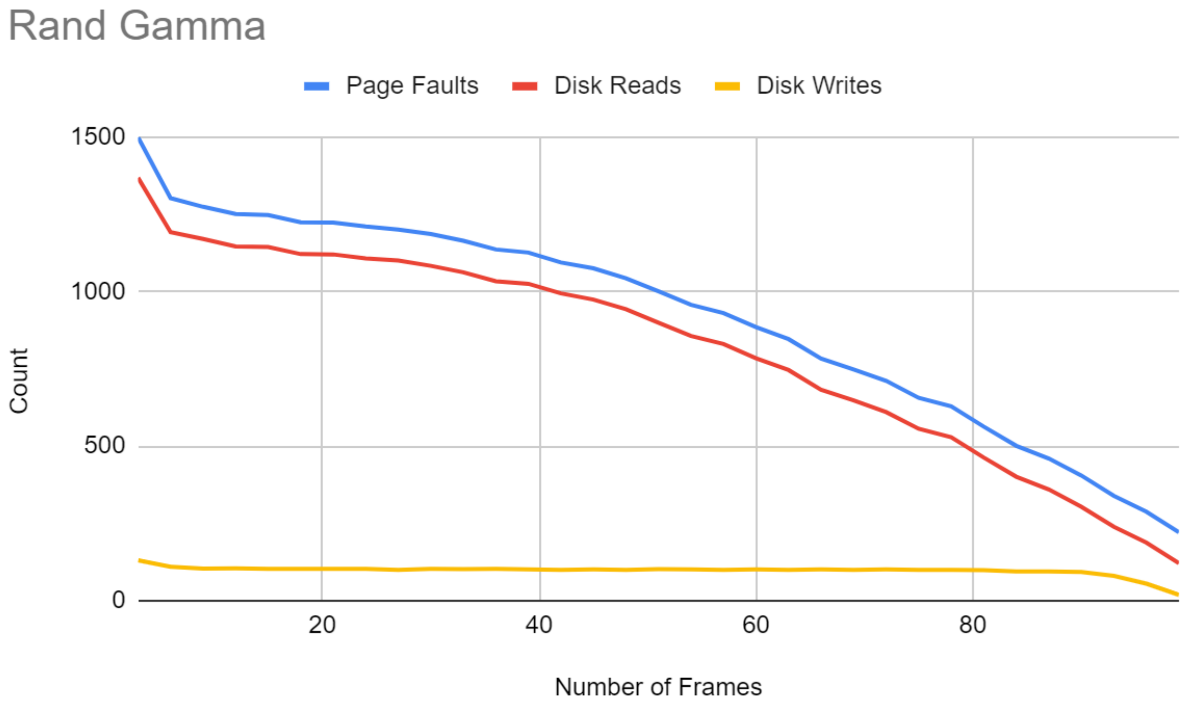


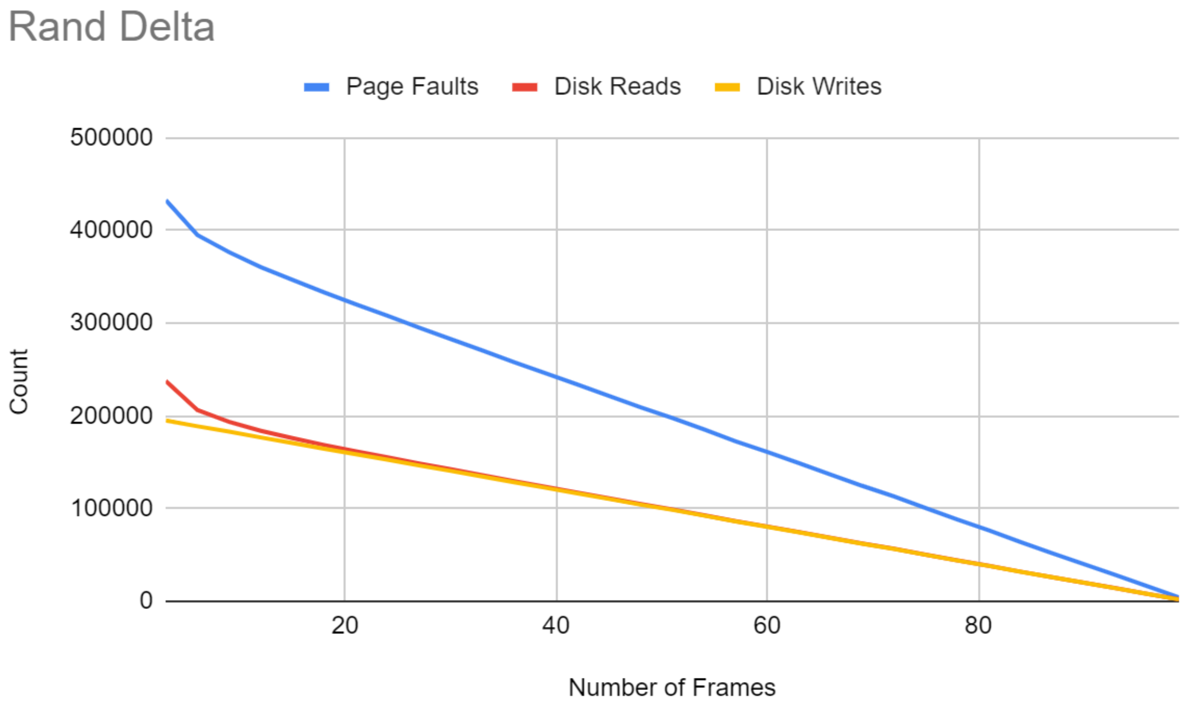
FIFO Delta

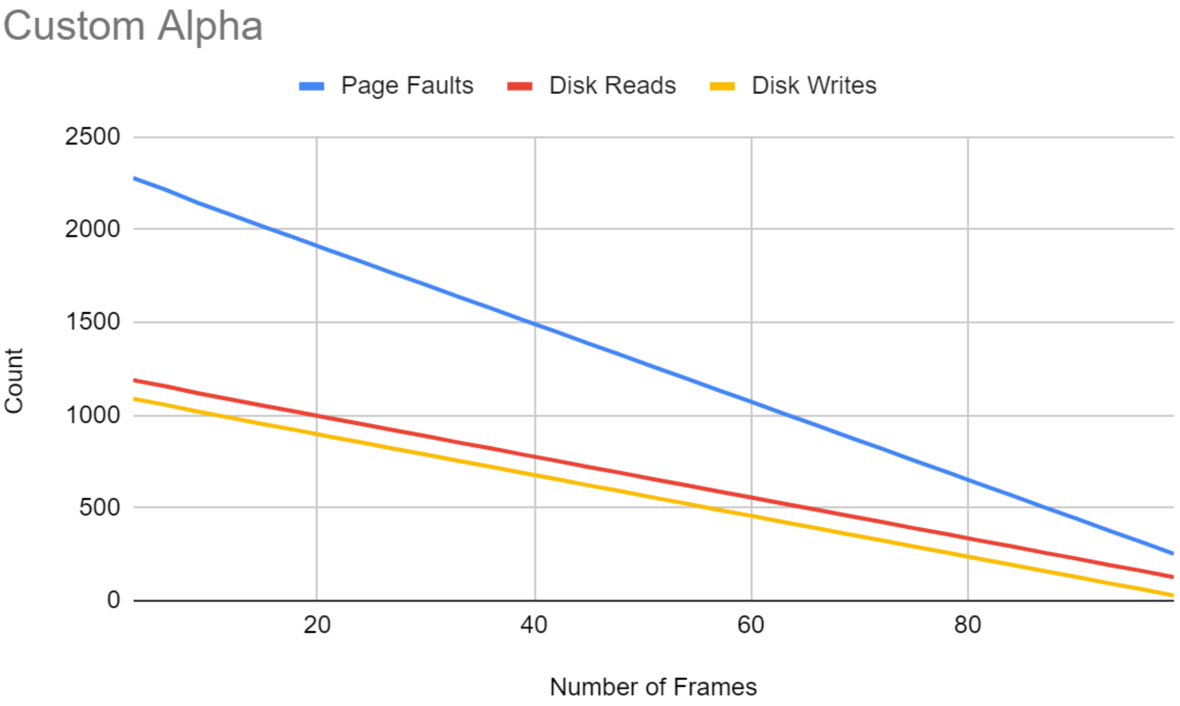


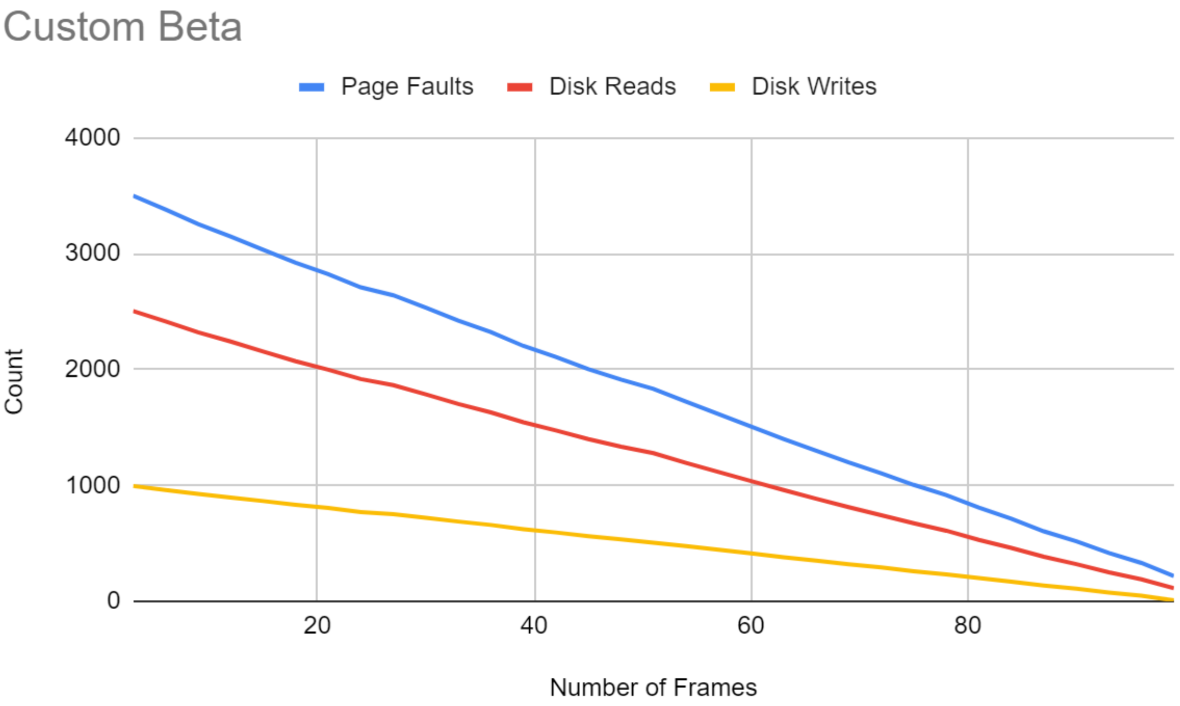

Rand Alpha
Page Faults
Disk Reads
Disk Writes
2500
2000
1500
1000
500
0
Count
20
40
60
80
Number of Frames

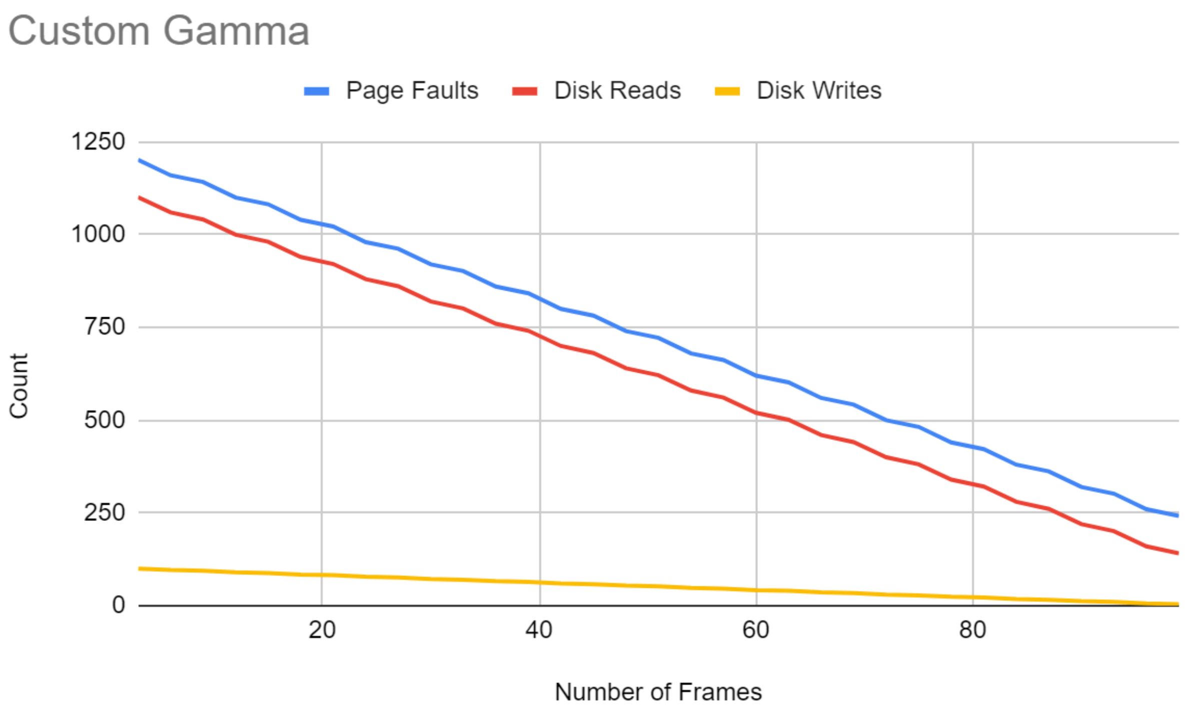

Rand Beta
Page Faults
Disk Reads
Disk Writes
3000
2000
1000
0
Count
20
40
60
80
Number of Frames

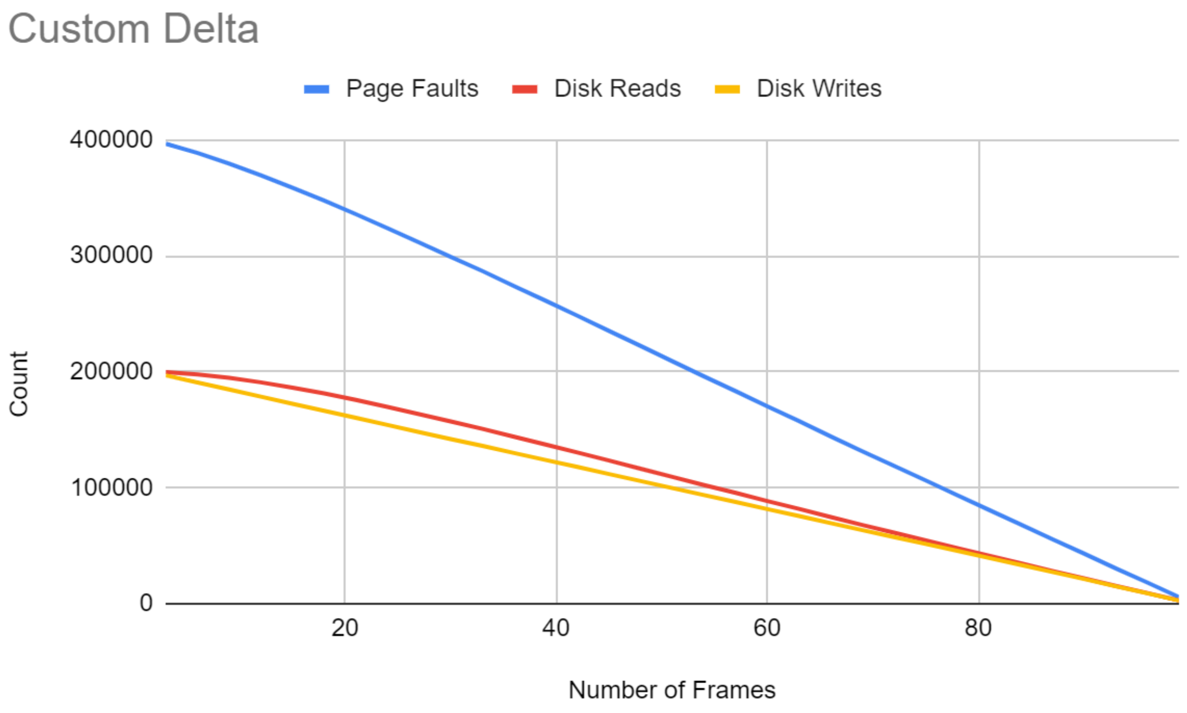

Rand Gamma
Page Faults
Disk Reads
Disk Writes
1500
1000
500
0
Count
20
40
60
80
Number of Frames


Rand Delta
Page Faults
Disk Reads
Disk Writes
500000
400000
300000
200000
100000
0
Count
20
40
60
80
Number of Frames


Custom Alpha
Page Faults
Disk Reads
Disk Writes
2500
2000
1500
1000
500
0
Count
20
40
60
80
Number of Frames


Custom Beta
Page Faults
Disk Reads
Disk Writes
4000
3000
2000
1000
0
Count
20
40
60
80
Number of Frames


Custom Gamma


Custom Delta

Explain the nature of the results. If one algorithm performs better than another under certain conditions, then point that out, explain the conditions, and explain *why* it performs better.

The performance of each algorithm is indicated by the number of page faults. A lower number of page faults indicates better performance. Overall, we can see that the delta program resulted in significantly more page faults, hovering around 400,000 total page faults for the FIFO, rand and custom algorithms. Gamma, on the other hand, resulted in significantly fewer page faults, hovering around 1250 to 1500 total faults for the FIFO, rand and custom algorithms. For the custom gamma combination, the number of page faults decreased as the number of frames increased, resulting in only about 250 page faults with 99 frames. This high performance is due to the nature of the custom algorithm. My custom algorithm evicts two pages at the same time, but only loads one new page, ultimately resulting in a fewer number of page faults.