Taylor Willittes

452 Proj4 Writeup

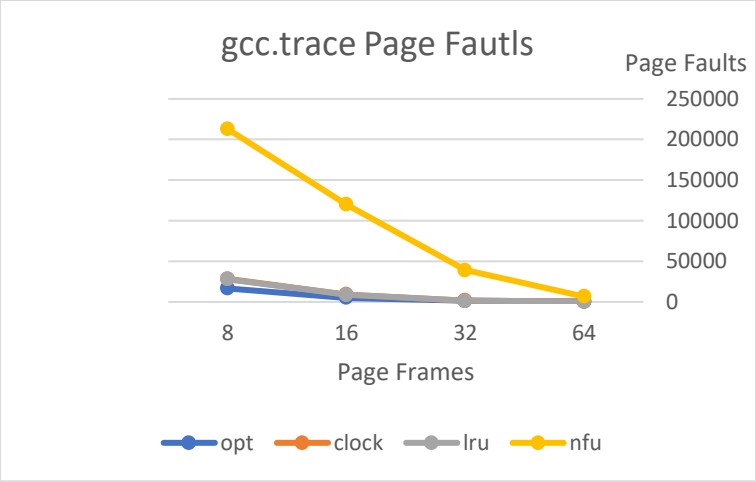
Opt is useful for us as a baseline of comparison even though its impossible to implement bc it requires knowledge about the future. Opt produces the perfect page fault count for the data given. This implies that if another algorithm produces results which are half as good as opt then designing a better algorithm leads to at most twice the performance.

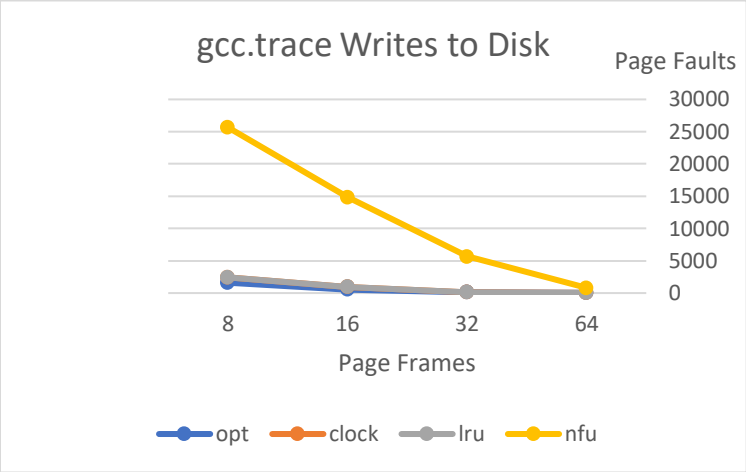
The clock alg attempts to optimize page eviction by evicting a page that has not been referenced recently. This alg will never evict a page we just used if an older page exists which means this alg avoids the worst-case scenario we talked about in class. Although if we just have 1 page frame then we would have to evict the same page making all 4 algs in this assignment have the same performance.

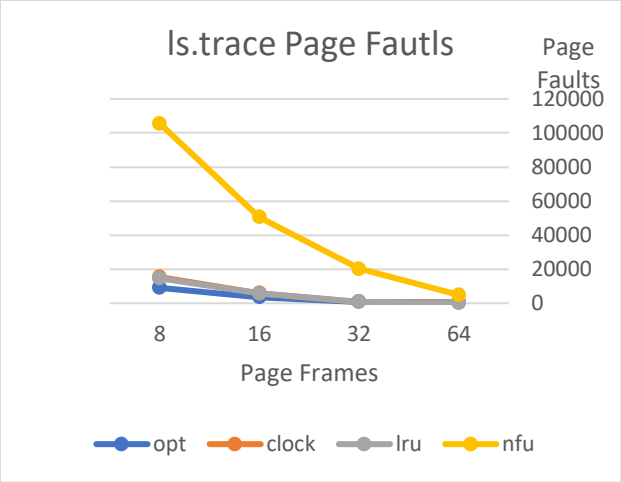
The lru alg will throw out a page that has been unused for the longest time. This algorithm is not cheap bc a linked list is needed to store data and must be updated on every memory reference. Searching for a page then deleting it and moving it to the front of the list makes this alg costly. Although this is not the worst alg performance wise as seen in the charts.

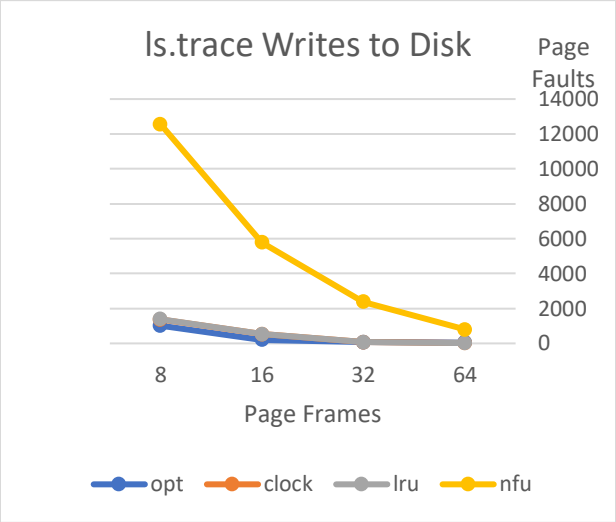
The nfu alg is very similar to the lru alg. When a page fault occurs, the page with the lowest referenced counter will be evicted. The issue with this algorithm is that it doesn't forget anything meaning if a page is used a lot in the first pass, it will still show that in the second pass. This alg performs the worst out of all 4 algs we implemented in this assignment as it is shown in the graphs.

As we can see from the graphs opt outperforms the other algorithms and nfu is the worst alg we implemented. The clock performs similar to lru, but it is easier to implement than lru. Overall, as the number of page frames increases, the number of page faults decreases. This lines up with what we learned bc we can store more pages in RAM leading to less evictions. This also gives more confirmation that these algorithms are implemented correctly.


gcc.trace Page Fautls
Page Faults
250000
200000
150000
100000
50000
0
8
16
32
64
Page Frames
opt
clock
lru
nfu


gcc.trace Writes to Disk
Page Faults
30000
25000
20000
15000
10000
5000
0
8
16
32
64
Page Frames
opt
clock
lru
nfu


ls.trace Page Fautls
Page Faults
120000
100000
80000
60000
40000
20000
0
8
16
32
64
Page Frames
opt
clock
lru
nfu


ls.trace Writes to Disk
Page Faults
14000
12000
10000
8000
6000
4000
2000
0
8
16
32
64
Page Frames
opt
clock
lru
nfu