Andrew Tran

CS 1550 Project 3 Write Up

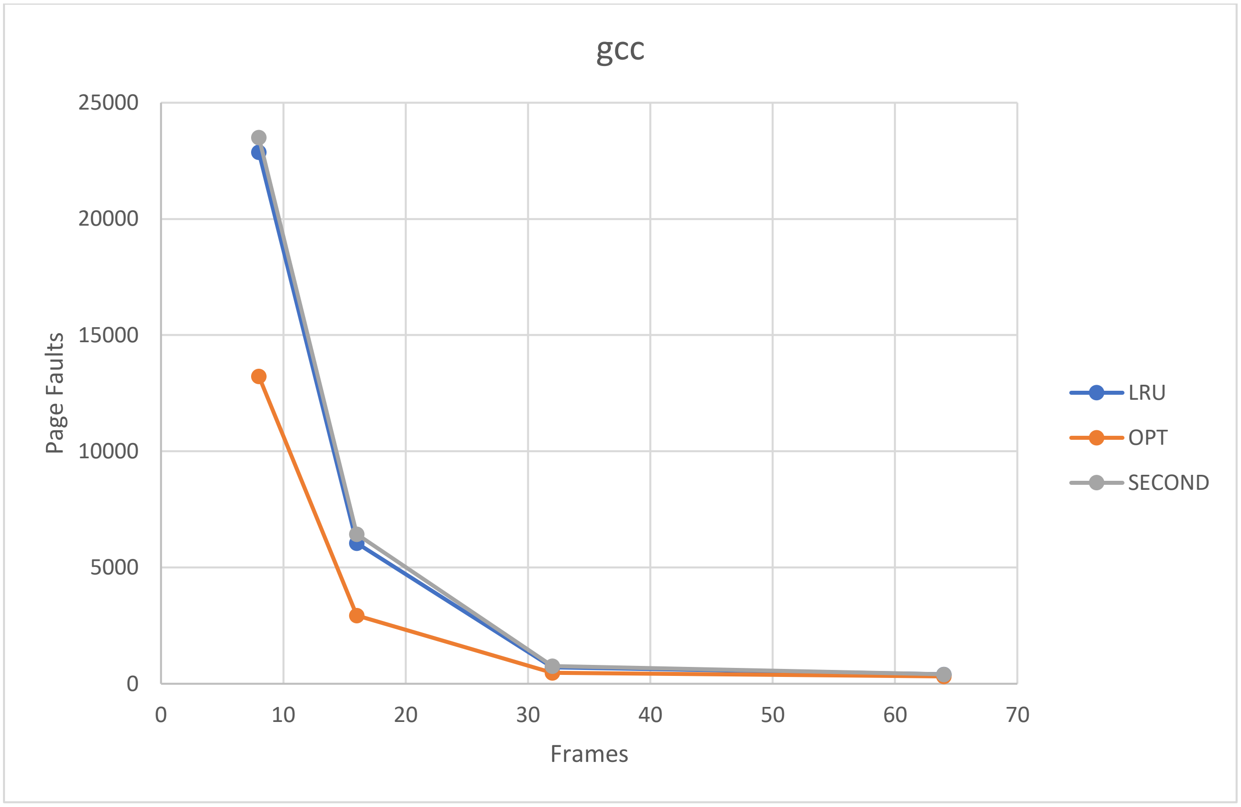
Part 1:

Gcc -

OPT	8	16	32	64
Faults	13227	2940	477	316

LRU	8	16	32	64
Faults	22866	6048	705	408

SECOND	8	16	32	64
Faults	23508	6443	764	408

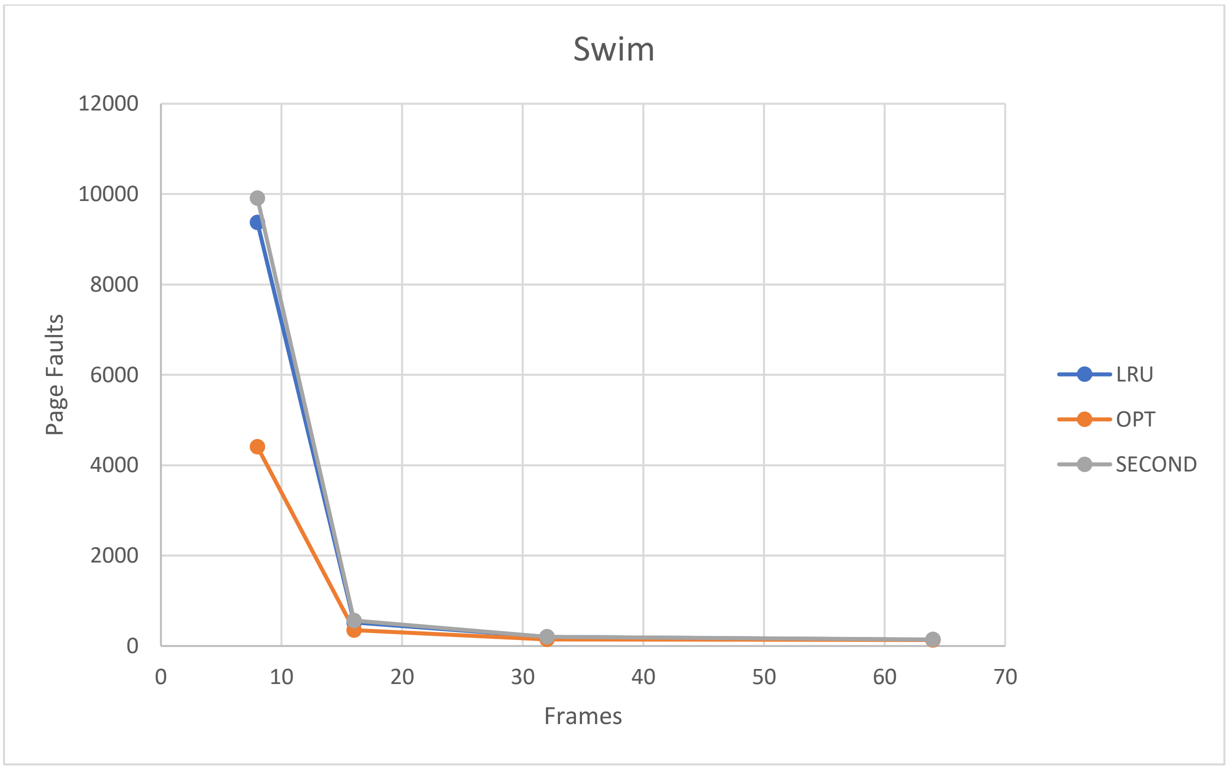


Swim –

OPT	8	16	32	64
Faults	4406	350	142	134

LRU	8	16	32	64
Faults	9372	522	199	138

SECOND	8	16	32	64
Faults	9911	566	201	144

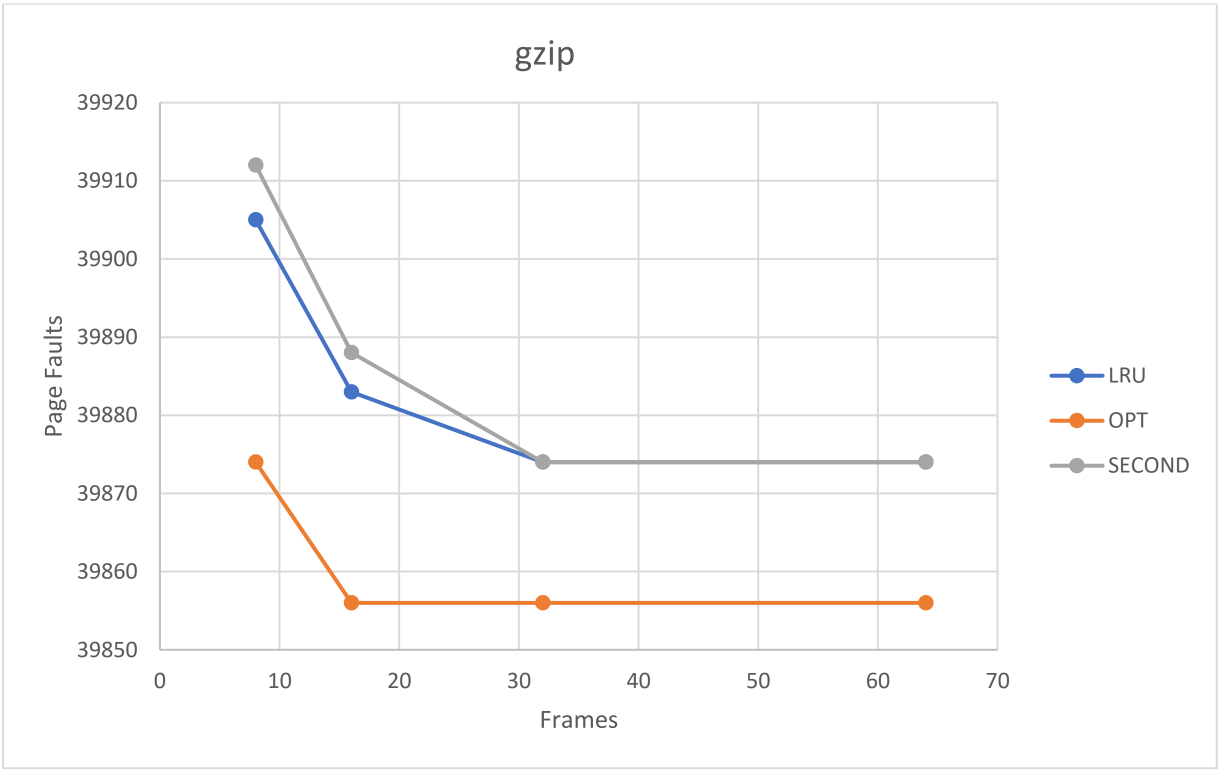


Gzip –

OPT	8	16	32	64
Faults	39874	39856	39856	39856

LRU	8	16	32	64
Faults	39905	39883	39874	39874

SECOND	8	16	32	64
Faults	39912	39888	39874	39874



Conclusion:

In all test cases, the LRU algorithm outperforms or does just as well as the second-chance algorithm even if just barely. So, LRU seems to be the better algorithm to use in an actual OS.

Part 2:

There were a few cases of Belady's Anomaly in all trace files such as in swim.trace, from 42→43 frames the number of faults goes from 168→170. All cases seem to occur when the rate at which the number of faults reaches 0. In gzip.trace, the number of faults pretty much stays constant at 39874 from 22 frames onward. There are a couple of instances where it will jump back up to 39875 though. In conclusion, it seems like adding more frames leads to diminishing returns and can sometimes even lead to slightly worse performance.

Part 3:

When the OPT mode is selected, the entire trace file is read before the simulation starts. The first read fills a hash table with all the information the simulation will need to know for choosing a page to evict. The hash table has keys representing every possible page number that can be represented in a 32-bit address space with a 4KB page size. The value for each key is an ordered linked list of integers representing the lines that page is referenced in in the trace file. When an address is read from the trace file during the simulation, the page number is calculated, and the corresponding linked list pops the head of the list. Since the list is ordered, the popped integer should always be the line number that was just read. When a page is being chosen for eviction, the simulation goes through every page currently in memory and checks the hash table for the next line number that will reference that page again in the future (the head of the linked list). If a page is found to not be referenced again (the size of its linked list is 0) it is immediately chosen for eviction. If all pages in memory will be referenced again, the page that is referenced the furthest in the future (has the highest integer value at the head of its linked list) will be chosen for eviction.

Since this algorithm uses a hash table and only checks the size and the head of the linked lists, all operations should run in constant time. The runtime of this algorithm should be bound by the number of frames and thus the number of pages that need to be checked every time an eviction is required.