Parker Hague

Professor Aakur

CS 5323

18 April 2022

Assignment 4 Analysis

Data Structure

For my implementation, I went with a simple data structure. I decided to use an array as the primary data structure to store my frames in the working set. I chose an array because it's simple to use and easy to implement. Though, I know that this isn't the most optimal data structure in terms of performance. The array has O(n) lookups whereas a data structure like a hash table has O(1) lookups. I think it's okay to take the performance hit in this scenario because our working set will never get bigger than a size of 5-10 realistically. The O(n) lookup would be a much bigger problem in a scenario where our n could be really big or if it's unknown and unpredictable.

Implementation

I implemented the FIFO replacement algorithm by using a pointer that keeps track of the location of the victim frame. The pointer starts at zero and increments to the next array location after each page miss in a circular manner.

LRU was very straightforward as well. To find my victim frame, I used an array to keep track of the index that each memory location in the working set was read at. I then looped over these indices and looked for the minimum value. The memory address with the minimum index would indicate that it was used the longest ago or least recently. This was the page that I removed.

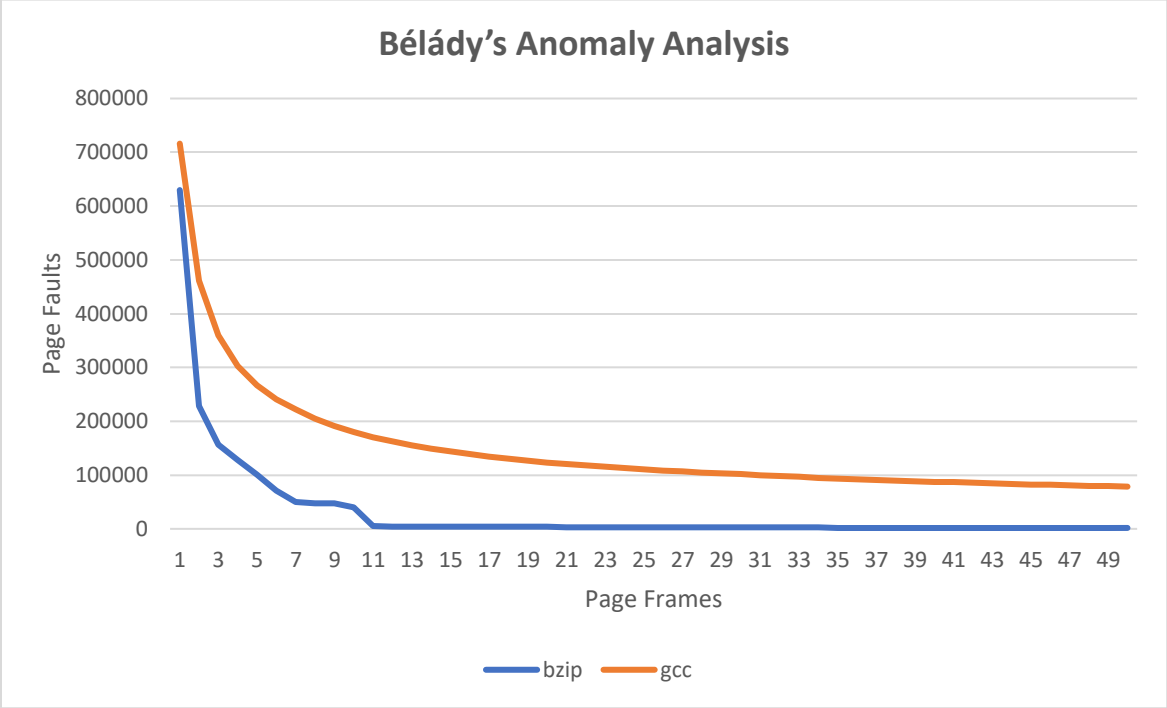
Analysis

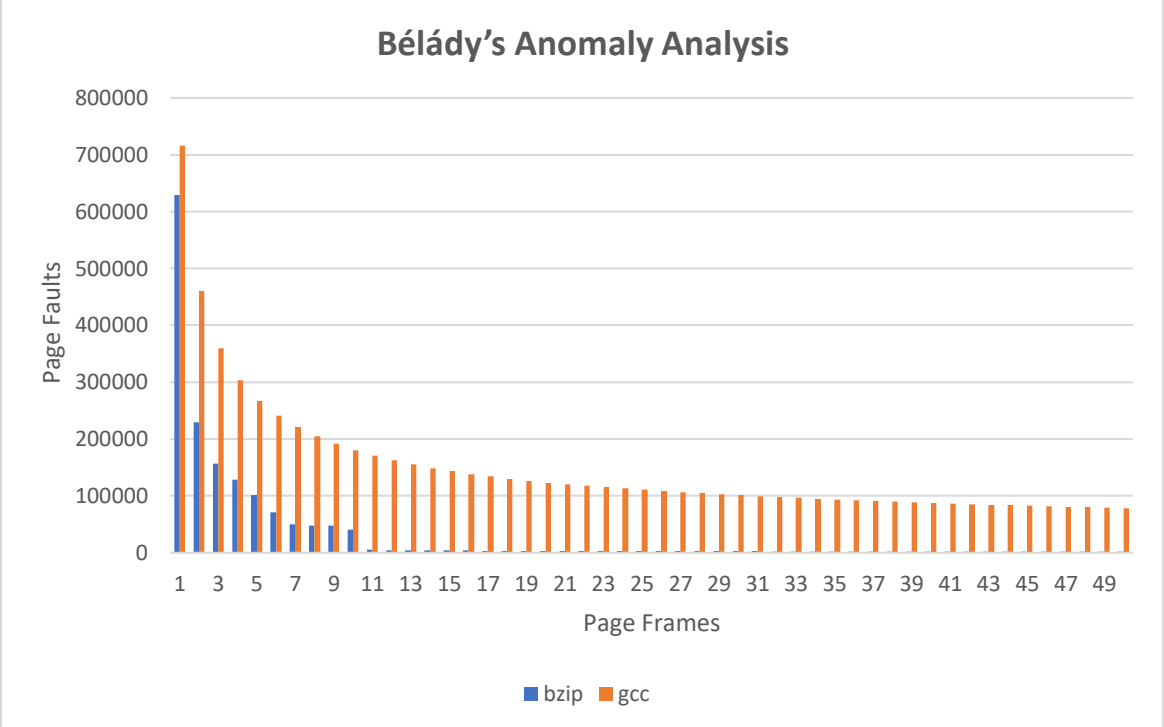
This program was interesting to test. As expected, the LRU algorithm greatly outperformed the FIFO algorithm. For example, when using four frames in the working set, the bzip.txt file had 27.9% fewer page faults when going from FIFO to LRU. The same test on the gcc.txt file saw a 19.5% decrease in page faults. These differences in page fault occurrences are quite large and really demonstrate that LRU is the superior algorithm.

Belady's Anomaly

I ran 100 tests across the files gcc.txt and bzip.txt. I tested each file for Belady's anomaly by running the program with 1-50 frames in the working set. Every time the number of frames in the working set was increased, the number of page faults decreased. After n=50 for each file, I was not able to observe Belady's anomaly occurring. The figures below show the number of page faults and how they decreased as the number of frames increased. They also give us an idea as to the ideal number of page frames in the working set. For example, for both files, we can tell that the number of page faults is dramatically decreasing until around the n=9-11 range. After that, the number of page faults begins to converge and plateau. This indicates that it might not be that efficient to have more than 10 frames or so in the working set. Of course, this may apply to only these trace files as each one is very unique.

# Frames	bzip	gcc
1	629737	716106
2	228838	460912
3	156574	359566
4	128601	302860
5	101120	266768
6	70658	240717
7	49815	221598
8	47828	205368
9	47232	191307
10	40385	179898
11	4907	170419
12	4581	162233
13	4361	155268
14	4105	148963
15	3927	143546
16	3820	138539
17	3695	134003
18	3629	129925
19	3534	126409
20	3480	123186
21	3398	120312
22	3346	117734
23	3292	115237
24	3236	112957
25	3188	110793
26	3133	108605
27	3066	106622
28	2668	104843
29	2627	103160
30	2602	101502
31	2568	99810
32	2497	98067
33	2334	96490
34	2259	94994
35	2162	93505
36	2085	92208
37	2048	90876
38	2015	89623
39	1987	88524
40	1883	87503
41	1855	86498
42	1829	85449
43	1813	84470
44	1772	83529
45	1743	82597
46	1728	81745
47	1719	80851
48	1700	80030
49	1689	79278
50	1667	78462


Bélády's Anomaly Analysis
800000
700000
600000
500000
400000
300000
200000
100000
0
Page Faults
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
Page Frames
bzip
gcc


Bélády's Anomaly Analysis
800000
700000
600000
500000
400000
300000
200000
100000
0
Page Faults
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
Page Frames
bzip
gcc