Name: Andrew Masih
Email: anm226@pitt.edu

VMSIM Analysis

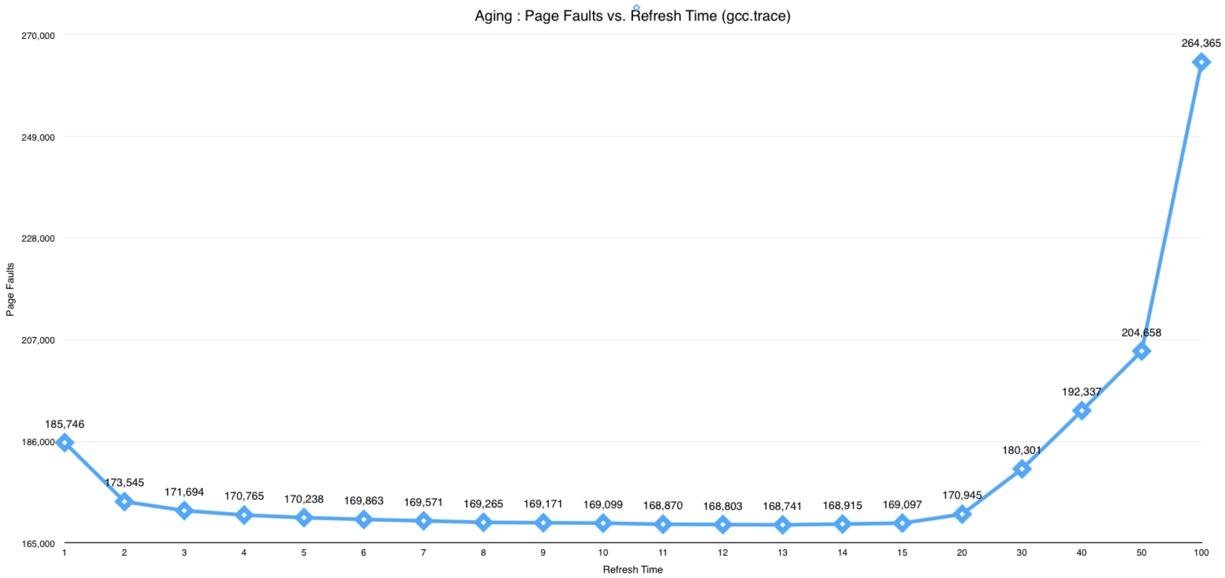
This document discusses the analysis done upon the Virtual Memory simulator constructed to test the four different page replacement algorithms. The algorithms analyzed include the Optimal Algorithm, the Aging Algorithm, the Least Recently Used Algorithm and the Second Chance Algorithm implemented using the clock style. The analysis is based on the use of the **gcc.trace** provided, all the data collected is using that file. So let's begin with this analysis.

First, I'd like to discuss how I found the best refresh rate for my Aging algorithm. Below is a table showing page faults resulting from different refresh times, where the number of frames was 8, and the number of memory accesses were 1,000,000.

gcc.trace				
Number Of Frames	Number of Memory Accesses	Refresh	Number Of Page Faults	Number of Disk Writes
8	1,000,000	1	185,746	26,730
8	1,000,000	2	173,545	24,234
8	1,000,000	3	171,694	23,765
8	1,000,000	4	170,765	23,641
8	1,000,000	5	170,238	23,407
8	1,000,000	6	169,863	23,310
8	1,000,000	7	169,571	23,174
8	1,000,000	8	169,265	23,066
8	1,000,000	9	169,171	22,955
8	1,000,000	10	169,099	22,853
8	1,000,000	11	168,870	22,821
8	1,000,000	12	168,803	22,675
8	1,000,000	13	168,741	22,663
8	1,000,000	14	168,915	22,586
8	1,000,000	15	169,097	22,619
8	1,000,000	20	170,945	22,683
8	1,000,000	30	180,301	23,717
8	1,000,000	40	192,337	24,485
8	1,000,000	50	204,658	25,385
8	1,000,000	100	264,365	31,032

Table 1

I'd like you to take notice the trend in the number of page faults as the refresh rate is increased. In the start, the page faults start out as large as 185,746. However as the number of refresh rate is increased, the number of page faults decrease, until we hit refresh rate of 14. There, the number of page faults again starts to increase, all the way up till refresh rate of 100. This trend can be seen even clearly in the graph below.



Using this data, I came to the conclusion of using **13** as my refresh rate since it gives me the least number of page faults.

Next I will compare the algorithms simulated in my program, a describe to you which one I believe would be the best implementation if I was developing my own Operating System. Below are some tables showing the data I collected by running the **gcc.trace** file on my simulator.

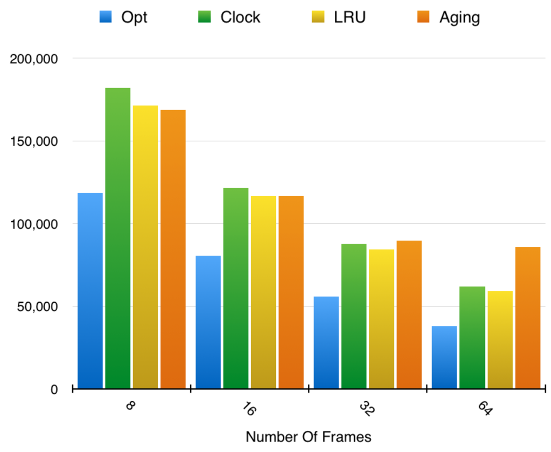
Optimal Algorithm		
Number of Frames	Number of Page Faults	Number of Disk Writes
8	118,480	15,031
16	80,307	11,316
32	55,802	8,274
64	38,050	5,730

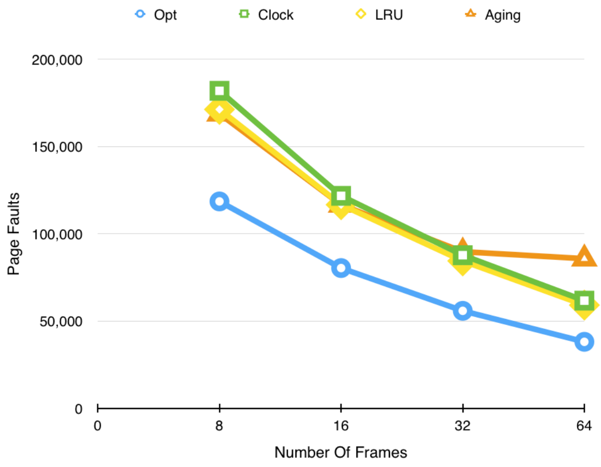
Clock Algorithm		
Number of Frames	Number of Page Faults	Number of Disk Writes
8	181,856	29,401
16	121,682	16,373
32	87,686	12,293
64	61,640	9,346

LRU Algorithm		
Number of Frames	Number of Page Faults	Number of Disk Writes
8	171,186	23,983
16	116,604	14,749
32	84,401	11,737
64	59,089	8,863

Aging Algorithm			
Number of Frames	Number of Page Faults	Number of Disk Writes	Refresh
8	168,741	22,663	13
16	116,552	14,652	13
32	89,681	12,087	13
64	85,698	11,678	13

As you can see Opt of course has the best results, but we know that we cannot implement an opt algorithm, because we do not know the future. Then we are left with LRU, Aging, and Clock algorithm. At first glance it may seem that Aging algorithm has the least amount of page faults but as the number of frames increases, it can be seen that the difference between the number of page faults in respect to number of frames is not that significant. Where LRU, and Clock seem to show a similar trend followed by the Opt algorithm. The trend that as the number of frames increases the number of page faults decreases, at a good rate. This trent can be more clearly seen in the graphs below.





In both the graphs it can be seen that LRU and Clock follow a similar trend to Opt, and both these algorithms are implementable. Now the choice is between LRU and Clock, and as it can be seen LRU does in fact give better performance than clock. I conclude that I would personally implement a LRU based page replacement algorithm. Now, one might say that this algorithm as implemented in the simulator using time stamps requires too much space, and is impractical. I would like to point out that at the speed the size of memory is increasing, and the price for that memory is decreasing, space will not be an issue at all in just little in the future. So that is why I would personally use the LRU algorithm for my OS, since it's closest to not just the number page faults of Opt, but also the trend that Opt shows as the number of frames increases.