PROJECT 2 REPORT

Made By: Jaylen Brown

Introduction

The following project was done to show how different page replacement policies effect overall system performance. To show this, my program takes in a set of a million traces which contain events of read and writes that were made during a program and with those traces my program will first determine which page number each data access belongs to and then it will iterate through each event acting like it's a page table to determine the efficiency of different page replacement policies. At the end of the evaluation, the program outputs the total reads and writes to disk which is an extremely slow process. With that said, the page replacement with the least of these is considered the most efficient in terms of speed or system performance. The three page replacement policies that will be tested with this program include First-in-First-Out (FIFO), Least-Recently-Used (LRU), and Segmented-FIFO (VMS).

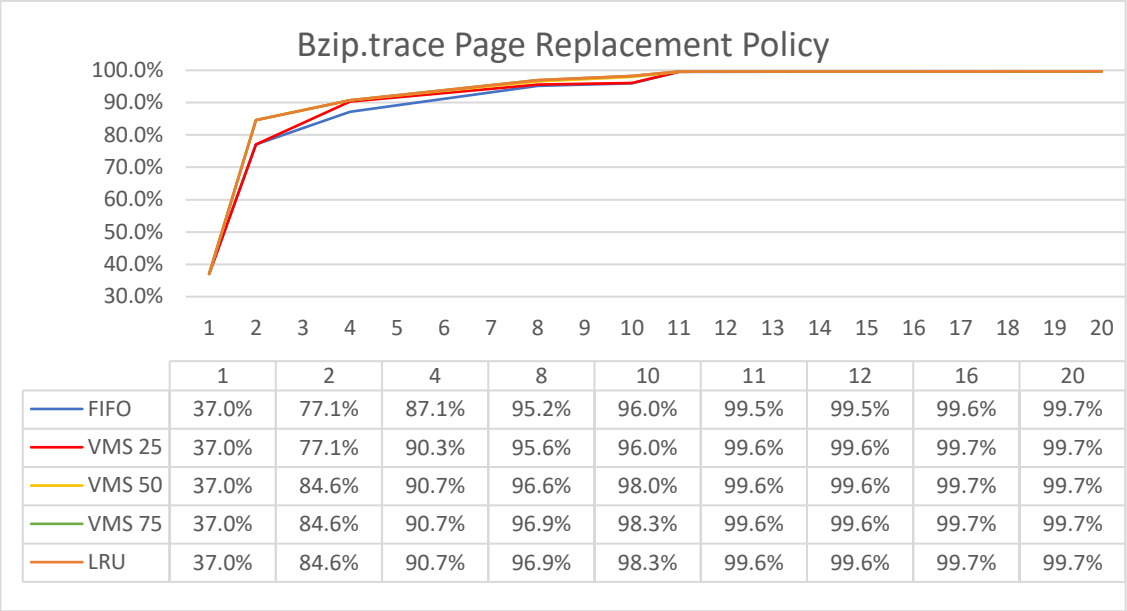
Methods

To begin my experiment, I first tested the bzip.trace file. For the test I inputted values of 2^m, keeping track of the number of disk reads being made until I noticed the disks reads weren't being impacted very much. I also tested the number of disk reads for a very small number of frames as well as a very high number of frames. After testing all input values, I typed out all the disk reads into an excel sheet and calculated the hit rate. I then created a graph that shows the hit rate vs number of frames. For the Sixpack.trace file I did the same steps as mentioned above, however, as you will notice in the results section, much higher input values were required to show the entire hit rate vs number of frames graph range.

When looking over the results I did notice that when the number of frames available is either 1 or an extremely high number then the hit rate is the same no matter the page replacement policy used. However, overall, the LRU page replacement policy gives the best performance. I also noticed that the segmentation FIFO page replacement policy or VMS gives performance that's better than FIFO but worse than LRU. With that said, if you set the secondary buffer percentage to 0 than the policy acts as FIFO while if the percentage is set to 100 then the policy acts as LRU.

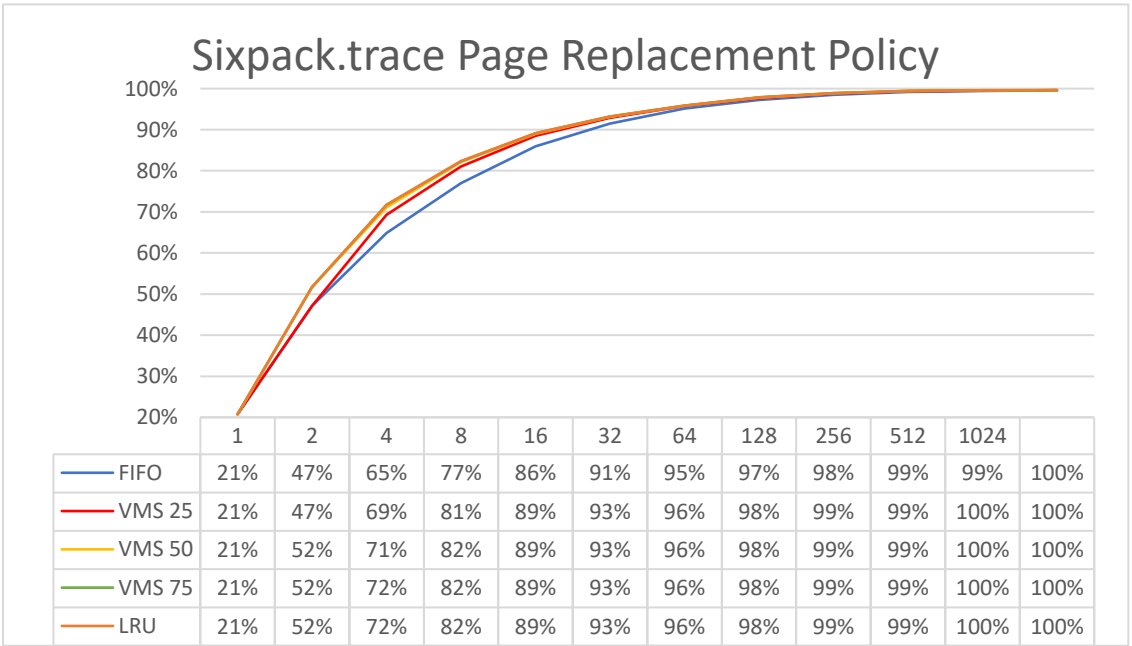
For these tests, each page/frame size was 4 KB (4096 bytes). With that said, if you increase the number of frames extremely high, you can find the total number of pages being used. For the bzip.trace file the total number of pages comes out to 317 which we can use this to find the total memory of the program needed. For the bzip file the total memory need is 1.23 MB or 1268 KB. Doing the same process for the Sixpack.trace file, the total number of pages comes out to 3890 which in terms of total memory is about 15.20 MB or 15560KB.

Results



	1	2	4	8	10	11	12	16	20
FIFO	37.0%	77.1%	87.1%	95.2%	96.0%	99.5%	99.5%	99.6%	99.7%
VMS 25	37.0%	77.1%	90.3%	95.6%	96.0%	99.6%	99.6%	99.7%	99.7%
VMS 50	37.0%	84.6%	90.7%	96.6%	98.0%	99.6%	99.6%	99.7%	99.7%
VMS 75	37.0%	84.6%	90.7%	96.9%	98.3%	99.6%	99.6%	99.7%	99.7%
LRU	37.0%	84.6%	90.7%	96.9%	98.3%	99.6%	99.6%	99.7%	99.7%

The following chart shows the hit rate vs. #frames used for the bzip.trace file. As shown above you can see that the LRU page replacement policy has either the same or better hit rate.



	1	2	4	8	16	32	64	128	256	512	1024	
FIFO	21%	47%	65%	77%	86%	91%	95%	97%	98%	99%	99%	100%
VMS 25	21%	47%	69%	81%	89%	93%	96%	98%	99%	99%	100%	100%
VMS 50	21%	52%	71%	82%	89%	93%	96%	98%	99%	99%	100%	100%
VMS 75	21%	52%	72%	82%	89%	93%	96%	98%	99%	99%	100%	100%
LRU	21%	52%	72%	82%	89%	93%	96%	98%	99%	99%	100%	100%

The following chart shows the hit rate vs. #frames used for the sixpack.trace file. As shown above you can see that the LRU page replacement policy has either the same or better hit rate.

Disk Reads					
#Frames	Page Replacement Policy				
	FIFO	LRU	VMS 25	VMS 50	VMS 75
1	629737	629737	629737	629737	629737
2	228838	154429	228838	154429	154429
4	128601	92770	96509	92954	92770
8	47828	30691	44006	33997	30685
10	40385	17120	39641	20370	17135
11	4907	4108	4390	4190	4113
12	4581	3907	4137	3969	3911
16	3820	3344	3445	3439	3351
20	3480	3001	3146	3053	3009
100	1026	906	949	921	914
2000	317	317	317	317	317

Shows the number of disk reads made for the bzip file per the number of frames used.

Disk Reads					
#Frames	Page Replacement Policy				
	FIFO	LRU	VMS 25	VMS 50	VMS 75
1	792379	792379	792379	792379	792379
2	529237	483161	529237	483161	483161
4	351810	282620	306866	288955	282620
8	230168	176496	189222	178933	176663
16	140083	108682	114938	109865	108816
32	85283	67747	71093	68616	67947
64	48301	41186	42408	41708	41357
128	27778	21090	23319	22216	21290
256	15440	11240	12084	11450	11275
512	8089	5823	6193	5874	5843
1024	5492	4466	4575	4488	4469
5000	3890	3890	3890	3890	3890

Shows the number of disk reads made for the sizpack file per the number of frames used.

Conclusion

Based on these experimental results, I have learned that the higher the size of the available memory the slower memory performance becomes. This was shown with the tables in the result sections. When looking at the hit rates you can see that the sixpack file which has a greater memory usage, requires a greater number of frames to achieve the high-performance rating. An example of this is looking at the hit rate for the LRU policy when the number of frames is 10 for the bzip file. For that number of frames, the hit rate is 98.3%. To achieve about the same performance in the sixpack file, the number of frames needed is about 128 frames which is considerably greater.

I have also learned that the LRU algorithm is overall a more efficient page replacement policy to use over both FIFO and VMS. However, when setting the VMS to a percentage of 50% we can achieve results close to the LRU algorithm which can be seen by looking at the tables provided in the results sections as well.