CS 620: Homework 12

Carmen St. Jean

November 29, 2012

1. *(3) Assume we have a demand-paged memory. The page table is held in registers. It takes 10 milliseconds to service a page fault if an empty frame is available or if the replaced page is not modified, and 25 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 60 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds?*

   The book defines effective access time as:

$$t = (1 - p) * ma + p * t_{pf}$$

   Where p is the probability of a page fault, ma is the memory access time, and t_{pf} is the page fault time. The average page fault time would be defined as:

$$t_{pf} = 0.60 * 25\,\text{ms} + 0.4 * 10\,\text{ms} = 19\,\text{ms} = 19000000\,\text{ns}$$

   Substituting this into the effective access time equation, we have:

$$\begin{aligned}
200\,\text{ns} &= (1 - p) * 100\,\text{ns} + p * 19000000\,\text{ns} \\
200\,\text{ns} &= 100\,\text{ns} - 100\,\text{ns} * p 19000000\,\text{ns} * p \\
100\,\text{ns} &= -100\,\text{ns} * p + 19000000\,\text{ns} * p \\
100\,\text{ns} &= 18999900\,\text{ns} * p \\
p &= 0.000005263
\end{aligned}$$

   Thus, the maximum acceptable page fault rate is 0.0005263%.

2. *(6) For the following reference string:*

   < a, b, c, d, c, d, c, d, d, a, d, E, d, c >.

   How many page faults would occur with the Working set model and the FIFO page replacement algorithm when MM has

   (a) *four frames and window size of four;*

a	b	c	d	c	d	c	d	d	a	d	E	d	c
a	b a	c b a	d c b a	c d b	d c	c d	d c	d c	a d c	d a	E d a	d E a	c d E
				a	b a	b a	b a	b a	b	c b	c	c	a
				*	*	*	*	*	*	*			

   (b) *three frames and window size of two*

a	b	c	d	c	d	c	d	d	a	d	E	d	c
a	b a	c b	d c	c d	d c	c d	d c	d	a d	d a	E d	d E	c d
		a	b	b	b	b	b	c b	c	c	a	a	E
				*	*	*	*	*		*		*	

All frames are initially empty, so your rst unique pages will all cost one fault each. Show both the working set and the replacement queue (stack) when each page is accessed.

3. *(12) Consider the following page reference string:*

   < 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6 >

   How many page faults would occur for the following replacement algorithms, assuming that MM has three frames? All frames are initially empty, so your rst unique p ages will all cost one fault each. Show the replacement queue at every access.

   (a) *Least Recently Used (LRU) replacement*

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3
		1	2	3	4	2	1	5	6	6	1	2	3	7	6	3	3	1	2
				*						*				*			*	*	

   15 page faults occured.

   (b) *Not Used Recently (NUR) replacement*

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1^1	1^1	1^1	4^1	4^1	4^1	4^0	6^1	6^1	6^1	6^1	6^0	7^1	7^1	7^1	7^0	1^1	1^1	1^1	1^0
	2^1	2^1	2^0	2^1	2^0	5^1	5^1	5^0	1^1	1^1	1^0	1^0	6^1	6^1	6^0	6^0	6^0	3^1	3^0
		3^1	3^0	3^0	1^1	1^1	1^0	2^1	2^1	2^1	3^1	3^1	3^1	3^1	2^1	2^1	2^1	2^1	6^1
				*						*				*			*		

   16 page faults occured.

   (c) *Most Recently Used (MRU) replacement*

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
	2	2	2	4	2	2	2	6	6	6	6	6	7	7	7	7	7	7	7
		3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
				*	*			*					*						

   16 page faults occured.

   (d) *Optimum replacement (OPT)*

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	6
	2	2	2	2	2	2	2	2	2	2	2	7	7	7	2	2	2	2	2
		3	4	4	4	5	6	6	6	6	6	6	6	6	6	1	1	1	1
				*	*			*	*	*		*	*	*			*	*	

   11 page faults occured.

4. *(8) (Question 9.28 of text book)*

   Consider a demand-paging system with the following time-measured utilizations:

CPU utilization	20%
Paging disk	97.7%
Other I/O devices	5%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Explain your answers.

(a) *install a faster CPU;*

No improvement; if the CPU is underutilized, then speeding it up will only exaggerate how under utilized it is.

(b) *install a bigger paging disk;*

No improvement; the amount of paging space is not he problem but the page retrieval speed.

(c) *increase the degree of multiprogramming;*

No improvement; this will worsen the problem by decreasing the number of frames per process and result in more thrashing.

(d) *decrease the degree of multiprogramming;*

Improvement; each process will have more frames for its use, which will result in less thrashing.

(e) *install more main memory;*

Improvement; more memory will result in more frames per program so there will be less thrashing.

(f) *install a faster hard disk or multiple controllers with multiple hard disks;*

No improvement; the problem is with the paging (therefore the memory) and not the disk.

(g) *add prepaging to the page-fetch algorithms;*

Improvement; the rate of page faults will be reduced.

(h) *increase page size.*

No improvement; if too much data are loaded into the pages, then there is some overhead, especially if the data are not needed later.

5. *(3) Consider the 2-dimensional array A: int A[][] = int[50][100]; where A[0][0] is stored at location 100, in a paged memory system with pages of size 100. A small process resides in page 0 (location 0 to 99) for manipulating the A matrix; thus every instruction fetch will be from page 0. Suppose MM has 6 page frames. How many page faults are generated by the following array initialization loop, using the LRU replacement policy?*

Note that page 0 is loaded into one of the page frames and must be constantly stored in memory while the program is running. Assume that each array element takes up 1 word of MM.

```
for (j = 0; j < 100; j++)
    for (i = 0; i < 50; i++)
        A[i][j] = 0;
```

One of the six page frames will be used for the small process in page 0, so only five will effectively be used for storing the array A. The array is most likely stored in row-major order. This means that a single page can hold a single row of array A. Therefore, the five page frames would hold ten rows of array A.

However, the nested for-loops will definitely be accessing the array A in column-major order. This means that every ten instructions will result in a page fault since the eleventh instruction would be trying access a row that is not loaded into the frames.

Since there are $50 * 100 = 5000$ instructions and a page fault once every ten instructions, there will be $5000/10 = 500$ page faults.

6. *(3) What is Bélády's anomaly? Demonstrate with an example.*

Bélády's anomaly is when more page faults occur after increasing the number of page frames while using FIFO.

a	b	c	d	a	b	E	a	b	c	d	E
a	b	c	d	a	b	E	E	E	c	d	d
	a	b	c	d	a	b	b	b	E	c	c
		a	b	c	d	a	a	a	b	E	E
							*	*			*

With three page frames, 9 page faults occur.

a	b	c	d	a	b	E	a	b	c	d	E
a	b	c	d	d	d	E	a	b	c	d	E
	a	b	c	c	c	d	E	a	b	c	d
		a	b	b	b	c	d	E	a	b	c
			a	a	a	b	c	d	E	a	b
				*	*						

With four page frames, 10 page faults occur, so it was slightly better when there were only three page frames.

I found this example at `http://en.wikipedia.org/wiki/Belady's_anomaly`.

7. *(1) What are stack replacement algorithms?*

   According to the textbook, stack replacement algorithms are algorithms which do not suffer from Bélády's anomaly, such as LRU.

8. *(4) Give an example of a specific page reference string and a specific memory size where the fault ratio of using FIFO is smaller than for LRU, or argue that no such example can exist.*

   Consider the FIFO example used in problem 6 with three pages. When this page reference string is used with three frames for LRU instead, there are 12 page faults:

a	b	c	d	a	b	E	a	b	c	d	E
a	b	c	d	a	b	E	a	b	c	d	E
	a	b	c	d	a	b	E	a	b	c	d
		a	b	c	d	a	b	E	a	b	c
							*				

With FIFO, only 9 page faults occured. Clearly, FIFO can have less page faults than LRU in some scenarios.