# **Luke Pepin - Homework 4**

# **Released: Apr 11, 2024**

# **Due: Apr 23, 2024, 11:59pm**

(5+5 points) Q1: Consider the following C program:

int X[N]; int step = M; // M is some predefined constant

for (int i=0; i =i+step)

X[i]=X[i]+1;

Assume that an integer is 1 Byte in size.

(a) If this program is run on a machine with a 4-KB page size and 64 entry TLB, what values of M and N will cause a TLB miss for every execution of the inner loop?

X[0] will cause a TLB miss and load for 1-4095 won’t be missed until X[4096], since 4096=4KB

**M = 4096/sizeof(int)**

Next for N and the 64 entry TLB, every 65th entry the main section needs to be removed

**N = 65 \* 4096/sizeof(int)**

(b) Would your answer in part (a) be different if the loop were repeated many times? Explain.

M in any case is 4096/sizeof(int) and N is 65 \* 4096/sizeof(int). Both remain unchanged given the calculations to determine when a TLB miss occurs depends on the page size and entry TLB not the number of loops. As a result, the values remain unchanged if the loop were repeated.

(5 points) Q2: Suppose that the virtual page reference sequence contains repetitions of long sequences of page references. For example, the sequence: 0,1,2,……,511,431,0,1,,,,,,511,431,0,1,…………consists of repetition of the sequence 0,1,…..,511,431.

If this sequence repeats 5 times (a total of 2565 page reference), how many page fault will you have if you use LRU, FIFO, and clock algorithm? Assume that you have 500 page frames allocated for this process.

**All Algorithms were not given enough information to determine exactly how many pages fault occurs. The following is my best guess and explanations with stated assumptions.**

LRU (least recently used):

Operates by removing the page that has been least recently accessed when a page fault occurs. With a sequence consisting of 2565-page references and only 500 available page frames, all 500 frames will incur page faults initially since we assume no pages are in physical memory before the sequence starts. Considering the estimated size of each "........" sequence is (2565-15)/4 = 613.5, the sequence "0,1,…..,511,431" contains a gap of references larger than the 500-page frames allocated. If no characters are repeated within the 500-page window, the LRU algorithm behaves similarly to the FIFO algorithm. This assumption is made since the sequence "0,1,2,...,511,431" is explicitly stated but no other sub-sequence is. As a result, almost every page reference is likely to result in a page fault, totaling **2565-page faults.**

FIFO (First In, First Out):

Operates by removing the page that has been in reference for the longest time. With the same assumptions as previously, this would result in the same result as LRU algorithm that doesn’t have any repeating page references within the 500-page frames allocated. Since both would operate on removing the oldest page reference in memory. Resulting in the same **2565-page faults.**

Clock algorithm:

Operates by maintaining a circular list of pages in memory and using a clock hand to determine which page to evict when a page fault occurs. Once again following the same assumptions the following occurs with the algorithm. The first 500 references incur page faults filling up the available page frames. Somewhere in the first “......” 0 and 1 are removed from the physical address for new pages since we assume 0 and 1 are not referenced twice in the first 502 characters resulting in their R-bit being 0 and being the oldest page references thus replaced. Next the remaining of the first “....” occurs the clock hand puts “511,431,0,1” into memory and another 500 references later in the second “......” sequence with no references those 4 pages are replaced since they were not referenced during their time in the clock algorithm. This continues for the remainer of the sequence resulting in once again **2565-page faults**.

(5 + 5 points) Q3: Assume that disk requests come in to the disk controller for cylinders 5, 10, 15, 55, 85, and 33, in that order. A seek takes 5 msec per cylinder move. How much seek time is needed to serve the requests if you use - In all cases, the arm is initially at cylinder 20. Show all the calculations.

a. First-come, first-serve algorithm

|current location – target location | \* time per cylinder move + next move...

|20-5|\*5 + |5-10|\*5 + |10-15|\*5 + |15-55|\*5 + |55-85|\*5 + |85-33|\*5 = 75+25+25+200+150+260

= **735 msec**

b. Closest cylinder next algorithm

|current location – target location | \* time per cylinder move + next move...

|20-15|\*5 + |15-10|\*5 + |10-5|\*5 + |5-33|\*5 + |33-55|\*5 + |55-85|\*5 = 25+25+25+140+110+150

= **475 msec**

(5 points) Q4: What is cylinder skew? Why is this useful?

Cylinder skew is a technique used in disk storage systems to optimize disk access times by intentionally skewing by offsetting the distance between the start of the last track of the previous cylinder. This ensures the head of the disk drive reaches the desired track slightly earlier than it would otherwise.

It is useful because it improves data access and reduces seek time optimizing disk performance. With the skew the disk head can start accessing data on the next track sooner this is done by reducing the time the disk head waits for track to get in position.

(5 + 5 points) Q5: Recall that both RAID level 0 and RAID level 1 use block-striping. The only difference is that RAID level 0 has no redundancy and RAID level 1 uses mirroring for redundancy.

(a) Could a RAID Level 1 organization achieve better performance for read requests (e.g., complete a read request faster) than a RAID Level 0 organization? If so, how?

Yes, a RAID Level 1 organization could result in better performance than a RAID Level 0 organization for read requests. This is because unlike level 0, level 1 using mirroring allows a read request to choose either disk to read from choosing a disk which is either closer or less busy, both of which can result in faster result times.

(b) Could a RAID Level 1 organization achieve better performance for write requests (e.g., complete a write request faster) than a RAID Level 0 organization? If so, how?

No, a RAID Level 1 organization could not result in better performance than a RAID Level 0 organization for write request. This is because mirroring level 1 requires each write request to happen twice. Increasing the likelihood a disk is busy, requiring more disk operations and as a result more time.

Assume that RAID 1 uses double the number of disks as RAID 0 (i.e., the number of primary disks is same in both cases).

(5 points) Q6: Let’s assume that a RAID can fail if two or more drive crashes within a short time interval. Suppose that the probability of one drive failing in a given hour is 0.05. What is the probability of a 10 drive RAID failing in a given hour?

P(X = 1) = 10 \* 0.05(1-0.05)^(10-1) = 0.3151

P(X = 0) = (1-0.05)^10 = 0.5987

P(X = 2) = 1 - P(X = 0) - P(X = 1)

P(X = 2) = 1 - 0.3151 - 0.5987 = **0.0862**

(5 points) Q7: Explain the Biba model for access control.

The Biba model of access control operates on the "no-read-up, no-write-down" policy, prioritizing data integrity by restricting the read and write operations between different integrity levels. Its primary goal is to prevent unauthorized modifications or corruption of data, ensuring the reliability of information flows within the system.