Homework 3

Jeremy Keys

11/30/17

Solution

Chapters 7, 8, 9, and 10

1)

> Consider a 64-bit address for a two-level paging system with a 10 KB page size. The outer page table has 2048 entries. How many bits are used to represent the secondlevel page table?

First, a 10KB page $= 10 * 1024 bytes = 10240$. $log_2(10240) = 13.32$**, so we need 14 bits to represent every byte in the page. (Side note: the page size could be 16KB without increasing the number of bits required to offset into every byte.) Next, the outer page table has 2048 entries;** $log_2(2048) = 11$**, so we need 11 bits to represent the outer page table. This leaves us with 64 bits, less 11 bits for the outer page number, less 14 bits for the page offset:** $64 - 11 - 14 = 39 bits$**.**

2)

> If the virtual address space supported is 2^64 bits, the page size is 1Kbyte, the size of the physical memory is 64Kbyte, the size of a PTE is two bytes, and the addressing is at the byte level, calculate the size of the page table required for both standard and inverted page tables.

First, for standard page tables: Each 1KB page contains 1024 bytes; indexing all bytes in a page (page offset) requires $log_2(1024) = 10$ **bits. The virtual address space supports** 2^{64} **bits, which implies it supports** 2^{61} **bytes. Since the ten lowest order bits are the (virtual) page offset, there are** $61 - 10 = 51$ **bits per page number. Thus, the virtual address space contains up to**2^{51} **pages. Finally, multiply the conveniently sized PTE of 2 with the number of pages,** $2^{51} * 2 = 2^{52}$**bytes.**

Segment	Base	Length
0	219	222
1	2300	121
2	1111	513
3	23	1234
4	445	3

Figure 1: Provided segment table

For inverted page tables: 64KB memory, each page is 1 KByte, thus there are 64KB/1KB = 64 bytes * 2 bytes/PTE = 128 bytes.

3)

Consider the following segment table. What are the physical addresses for the following logical addresses? 0, 130; 1,150; 2,500; 3,400; 4,12

1. <0,130> = 219+130 = 349
2. <1,150> = 2300+150 = 2450; invalid logical address, trap to kernel
3. <2,500> = 1111 + 500 = 1611
4. <3,400> = 23 + 400 = 423
5. <4,12> = 445 + 12 = 457; invalid logical address, trap to kernel

4)

Suppose we have the following page accesses: 1 2 3 4 2 3 4 1 2 1 1 3 1 4 and that there are three frames within our system. Using the FIFO and LRU replacement algorithm, show the paging sequence and identify the number of page faults.

My solution is attached as an image. I got 8 page faults for both FIFO and LRU. Note that I accidentally omitted the references to the page sequence 2-3-4 (with arrows) on the LRU run. Luckily, all of those pages are in a frame during that sequence, so instead of erasing and adding them, I just drew arrows to represent that they exist.

5)

Describe a scenario in which FIFO page replacement results in high paging overhead, but LRU does better.

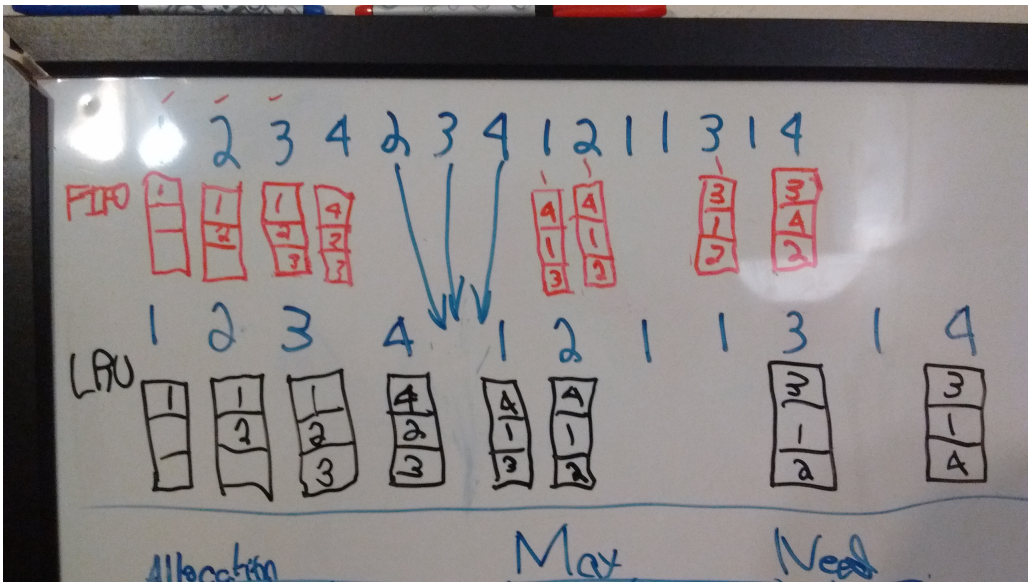

Figure 2: The FIFO and LRU runs. Note that the page sequence 2,3,4 in the LRU were inadvertently omitted but did not change the answer, hence the arrows.

A FIFO page replacement policy will result in high paging overhead in cases where some pages are accessed heavily and some are accessed infrequently. With FIFO, a heavily used page that arrives earlier than a lightly used page will be evicted, when that is undesirable: it would be better to keep the "important" (highly accessed) pages in a page frame, rather than paging to disk and immediately triggering a page fault. An example might be a program that has several page-sized arrays (more than the amount of frames in the system), but some arrays are iterated over/accessed more frequently while the others are comparatively unused. In that case, the highly used page(s) will be swapped when a rarely used page is required (triggering a page fault), even though the highly used page is likely to be requested soon and trigger a page swap.

6)

Consider the following snapshot of a system. Using banker's algorithm: a) show content of Need matrix; b) show that the system is in a safe state by listing the order in which processes can be executed without producing a deadlock. Also, if request for P4 arrives at (0, 1, 0, 0), c) show the updated Need, Allocation and Available matrices. If the process request can be granted immediately, show the safe sequence. If not, list the processes that are possible in a

	Allocation					Max			
	A	B	C	D		A	B	C	D
P_0	2	0	0	1		3	4	1	2
P_1	0	2	0	1		1	4	2	3
P_2	1	1	1	1		2	3	4	2
P_3	1	1	1	1		2	5	5	3
P_4	1	0	1	0		3	5	3	1

Figure 3: The system snapshot

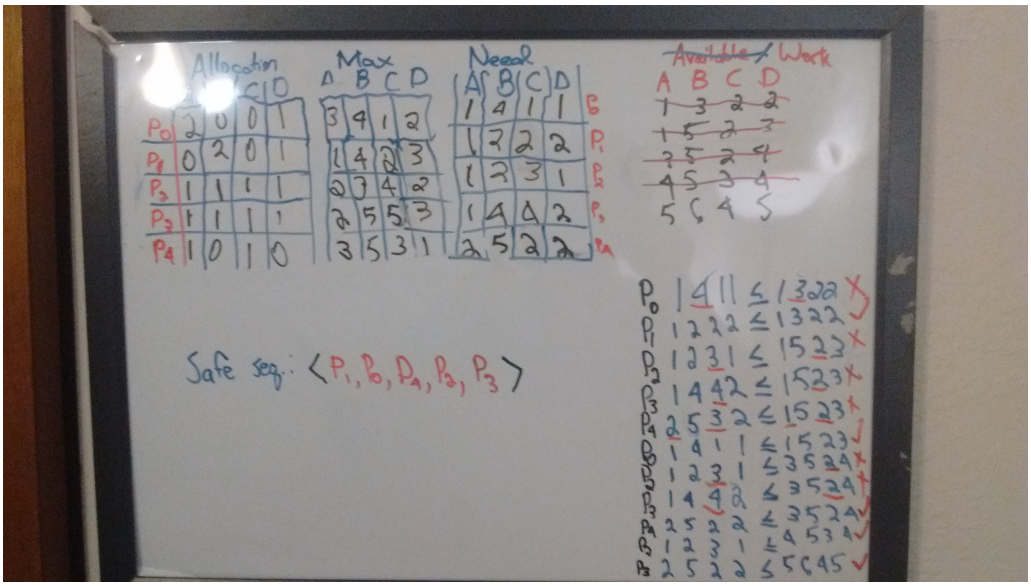

Figure 4: Safe page sequence as found by using Banker's Algorithm

deadlock. **Initially, P0 availability (1, 3, 2, 2).**

My solution is attached. For the first part, the safe sequence is <P1, P0, P4, P2, P3>. After servicing P4's request, the safe sequence becomes <P1, P0, P2, P4, P3>.

7)

Write buffering is a technique for writing data to the write buffer first and flushing data to the disk later. The write buffer is allocated in the memory, which has a faster access time than the disk. Allocating a large write buffer in the memory may help improve write performance, but can lead to lack of memory required to run applications. Assume the peak transfer rate of disk Rpeak = 200 MB/s, the positioning overhead (i.e., rotation delay + seek time) Tpostition = 10 ms, the size of the data to write D = 500 MB. If we want to

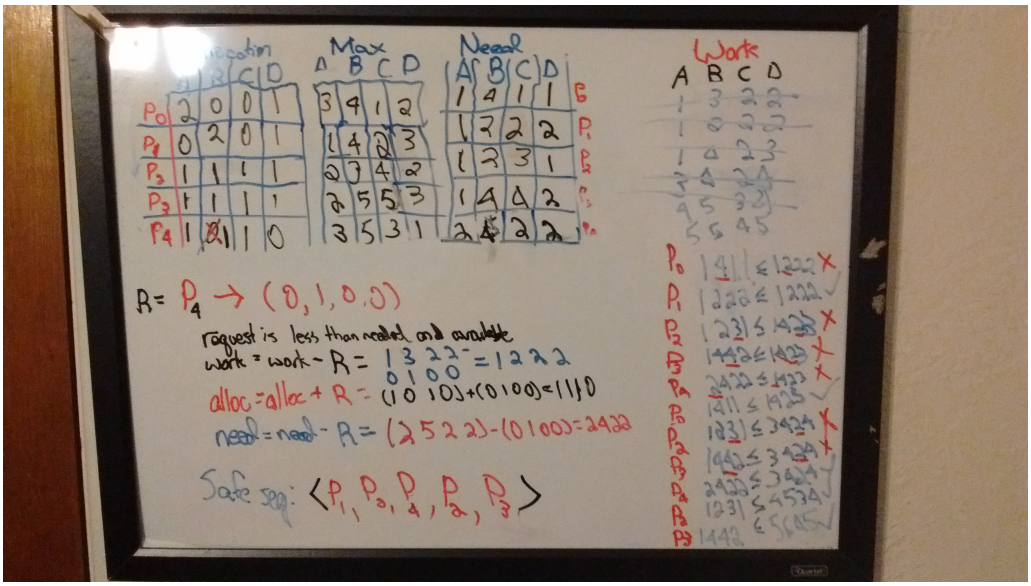

Figure 5: Safe page sequence, after servicing P4's request, as found by using Banker's Algorithm

achieve an effective transfer rate that equals to 90% peak transfer rate, how many data should be buffered?

I don't understand the problem the way it's formulated. I'd say the size of the buffer should be equal to the size of the effective rate, which is 200MB * 90% = 180MB.

8)

Consider a disk queue holding requests to the following cylinders in the listed order: 122, 45, 2, 21, 44, 185, 100, 822. What is the order that the requests are serviced, assuming the disk head is at cylinder 88 and moving upward through the cylinders? Using a) SCAN, b) C-SCAN and c) SSTF.

1. SCAN: 100, 122, 185, 822, 45, 44, 21, 2
2. C-SCAN: 100, 122, 185, 822, 2, 21, 44, 45
3. SSTF: 100, 122, 185, 45, 44, 21, 2, 822