Section 8.5: Extra Section Potpourri

CS 162

October 23, 2020

Contents

Note: These questions are usually included on section worksheets in previous semesters. In an effort to make section worksheets more manageable, we have omitted these questions from previous worksheets because these would be questions that TAs would likely have not had time for in a normal section. These questions are still in scope for the midterm, but are generally not as emphasized as opposed to other topics.

1 Advanced Scheduling

In what sense is CFS completely fair?

How do we easily implement Lottery scheduling?

Is Stride scheduling prone to starvation?

In Stride scheduling, if a job is more urgent, should it be assigned a larger stride or a smaller stride?

2 Page Fault Handling for Pages Only On Disk

The page table maps VPN to PPN, but what if the page is not in main memory and only on disk? Think about structures/bits you might need to add to the page table/OS to account for this. Write pseudocode for a page fault handler to handle this.

3 Inverted Page Tables

3.1 Inverted Page Table

Why IPTs? Consider the following case:
- 64-bit virtual address space
- 4 KB page size
- 512 MB physical memory

How much space (memory) needed for a single level page table? Hint: how many entries are there? 1 per virtual page. What is the size of a page table entry? access control bits + physical page #.

How about multi level page tables? Do they serve us any better here?

What is the number of levels needed to ensure that any page table requires only a single page (4 KB)?

3.2 Linear Inverted Page Table

What is the size of of the hashtable? What is the runtime of finding a particular entry?

Assume the following:
- 16 bits for process ID
- 52 bit virtual page number (same as calculated above)
- 12 bits of access information

3.3 Hashed Inverted Page Table

What is the size of of the hashtable? What is the runtime of finding a particular entry?

Assume the following:

- 16 bits for process ID
- 52 bit virtual page number (same as calculated above)
- 12 bits of access information

4 Second Chance List Algorithm

Suppose you have four pages of physical memory: 00, 01, 10, 11 and five pages of virtual memory: 000, 001, 010, 011, 101. Run the second chance list algorithm, with two physical pages delegated to the active list and two physical pages delegated to the second chance list.

The access pattern (assume we write to these pages) for virtual pages is as follows:

Page	000	001	010	011	000	101

The page table looks like this at the start of the algorithm.

Virtual Page	Physical Page	Extra
000		PAGEOUT
001		PAGEOUT
010		PAGEOUT
011		PAGEOUT
101		PAGEOUT

The 'extra' bits should read 'RW', 'INVALID', 'FIFO: 0', 'LRU: 0' where the bit for FIFO / LRU is 0 for more recent and 1 for less recent.

4.1 After Writes to '000', '001'

What does the table look like after these first two writes?

Virtual Page	Physical Page	Extra
000		
001		
010		
011		
101		

4.2 After Writes to ’010’, ’011’

What does the table look like after the next two writes?

Virtual Page	Physical Page	Extra
000		
001		
010		
011		
101		

4.3 After Write to ’000’

What happens when you write to virtual page 000? What does the table look like after this write?

Virtual Page	Physical Page	Extra
000		
001		
010		
011		
101		

4.4 After Write to ’101’

What happens when you write to virtual page 101? What does the table look like after this write?

Virtual Page	Physical Page	Extra
000		
001		
010		
011		
101		