CIS 415 – Operating Systems

Homework Assignment 2
Fall 2016 – Prof. Sventek

Due at 5:00pm on Tuesday, 25 October 2016

All questions must be answered by you without outside assistance. **Submission is via Canvas.** You may submit either a plain text (.txt) or a PDF (.pdf) file. Succinct, concise answers to the questions are preferable to long, rambling ones.

Textbook Questions (50 points)

1. OSC 4.14: A system with two dual-core processors has four processors available for scheduling. A CPU-intensive application is running on this system. All input is performed at program start-up, when a single file must be opened. Similarly, all output is performed just before the program terminates, when the program results must be written to a single file. Between start-up and termination, the program is entirely CPU-bound. Your task is to improve the performance of this application by multithreading it. The application runs on a system that uses the one-to-one threading model (each user thread maps to a kernel thread).
    a. How many threads will you create to perform the input and output? Explain. (3 points)
    It only makes sense to create as many threads as there are blocking system calls, as the threads will be spent blocking. Creating additional threads provides no benefit. Thus, it makes sense to create a single thread for input and a single thread for output.
    b. How many threads will you create for the CPU-intensive portion of the application? Explain. (3 points)
    Four. There should be as many threads as there are processing cores. Fewer would be a waste of processing resources, and any number > 4 would be unable to run.
2. OSC 4.18: Consider a multicore system and a multithreaded program written using the many-to-many threading model. Let the number of user-level threads in the program be greater than the number of processing cores in the system. Discuss the performance implications of the following scenarios. (9 points)
    a. The number of kernel threads allocated to the program is less than the number of processing cores.
    b. The number of kernel threads allocated to the program is equal to the number of processing cores.
    c. The number of kernel threads allocated to the program is greater than the number of processing cores but is still less than the number of user-level threads.
    When the number of kernel threads is less than the number of processors, then some of the processors would remain idle since the scheduler maps only kernel threads to processors and not user-level threads to processors. When the number of kernel threads is exactly equal to the number of processors, then it is possible that all of the processors might be utilized simultaneously.However,when a kernel-thread blocks inside the kernel (due to a page fault or while invoking system calls), the corresponding processor would remain idle. When there are more kernel threads than processors, a blocked kernel thread could be swapped out in favor of

another kernel thread that is ready to execute, thereby increasing the utilization of the multiprocessor system.

3. OSC 6.6: Suppose that a short-term CPU scheduling algorithm favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound processes and yet not permanently starve CPU-bound programs? (6 points)
It will favor the I/O-bound programs because of the relatively short CPU burst request by them; however, the CPU-bound programs will not starve because the I/O-bound programs will relinquish the CPU relatively often to do their I/O

4. OSC 6.11: Discuss the following pairs of scheduling criteria conflict in certain settings (9 points):
    a. CPU utilization and response time
    CPU utilization is increased if the overheads associated with context switching is minimized. The context switching overheads could be lowered by performing context switches infrequently. This could, however, result in increasing the response time for processes.
    b. Average turnaround time and maximum waiting time
    Average turnaround time is minimized by executing the shortest tasks first. Such a scheduling policy could, however, starve long-running tasks and thereby increase their waiting time.
    c. I/O device utilization and CPU utilization
    CPU utilization is maximized by running long-running CPU-bound tasks without performing context switches. I/O device utilization is maximized by scheduling I/O-bound jobs as soon as they become ready to run, thereby incurring the overheads of context switches.

5. Real-time scheduling
    a. Rate monotonic (RM) is a well-known, fixed-priority scheduling algorithm for periodic tasks. Describe how RM works, and why it is classified as a fixed-priority algorithm. (4 points)
    RM assigns priorities to tasks based on their periods – the shorter the period, the higher the priority. Since the priority only depends upon the rate at which jobs are initiated, all jobs in the task are assigned the same priority, making it a fixed priority algorithm.
    b. You are given a set of independent, periodic tasks: $T_1 = (0, 4, 1)$, $T_2 = (0, 8, 2)$, $T_3 = (0, 20, 2)$.
        i. There exists a feasible schedule for these tasks using RM. Why? (4 points)
        The total utilization is 0.25 + 0.25 + 0.1 = 0.6; if the total utilization for n tasks is less than $n(2^{1/n} - 1)$, the schedule is feasible. Since $0.6 < 0.771$, there is a feasible schedule.
        ii. Assume that you are asked to add another task, $T_4 = (0, \pi, 1)$, such that this new system, $\{T_1, T_2, T_3, T_4\}$ has a feasible schedule using RM. You must determine the value of π. A colleague asserts that $\pi = 5$ will work. Do you agree? Why or why not? (12 points)
        If $\pi == 5$, the total utilization is 0.25 + 0.25 + 0.1 + 0.2 = 0.8; if the total utilization for n tasks is less than $n(2^{1/n} - 1)$, the schedule is feasible. Since

0.8 > 0.757, we have to simulate a schedule through a hyper-period to see if a feasible schedule exists.

Time	Ready to run queue	Scheduled
0	$J_{1,1}$;$J_{4,1}$;$J_{2,1}(2)$;$J_{3,1}(2)$	$J_{1,1}$
1	$J_{4,1}$;$J_{2,1}(2)$;$J_{3,1}(2)$	$J_{4,1}$
2	$J_{2,1}(2)$;$J_{3,1}(2)$	$J_{2,1}(2)$
4	$J_{1,2}$;$J_{3,1}(2)$	$J_{1,2}$
5	$J_{4,2}$;$J_{3,1}(2)$	$J_{4,2}$
6	$J_{3,1}(2)$	$J_{3,1}(2)$
8	$J_{1,3}$;$J_{2,2}(2)$	$J_{1,3}$
9	$J_{2,2}(2)$	$J_{2,2}(2)$
10	$J_{4,3}$;$J_{2,2}(1)$	$J_{4,3}$
11	$J_{2,2}(1)$	$J_{2,2}(1)$
12	$J_{1,4}$	$J_{1,4}$
13	empty	nothing
15	$J_{4,4}$	$J_{4,4}$
16	$J_{1,5}$;$J_{2,3}(2)$	$J_{1,5}$
17	$J_{2,3}(2)$	$J_{2,3}(2)$
19	empty	nothing
20	$J_{1,6}$; $J_{4,5}$; $J_{3,2}(2)$	$J_{1,6}$
21	$J_{4,5}$; $J_{3,2}(2)$	$J_{4,5}$
22	$J_{3,2}(2)$	$J_{3,2}(2)$
24	$J_{1,7}$; $J_{2,4}(2)$	$J_{1,7}$
25	$J_{4,6}$; $J_{2,4}(2)$	$J_{4,6}$
26	$J_{2,4}(2)$	$J_{2,4}(2)$
28	$J_{1,8}$	$J_{1,8}$
29	empty	nothing
30	$J_{4,7}$	$J_{4,7}$
31	empty	nothing
32	$J_{1,9}$; $J_{2,5}(2)$	$J_{1,9}$
33	$J_{2,5}(2)$	$J_{2,5}(2)$
35	$J_{4,8}$	$J_{4,8}$

36	$J_{1,10}$	$J_{1,10}$
37	empty	nothing
40	$J_{1,11}$ $J_{4,9}$;$J_{2,6}(2)$;$J_{3,3}(2)$	

After simulating through an entire hyper-period (40), we see that we have a feasible schedule when T_4 is added with $\pi = 5$.

Note: Like all assignments in this class **you are prohibited from copying any content from the Internet or sharing ideas, code, configuration, text or anything else or getting help from anyone in or outside of the class, except where noted**. Consulting online sources is acceptable, but under no circumstances should anything be copied. Failure to abide by this requirement will result in sanctions ranging from zero on the assignment to dismissal from the class.