Cole Vikupitz

CIS 415

28 April 2017

Assignment 2

Textbook Questions

1. OSC 4.14: *A system with two dual-core processors has four processors available for scheduling. A CPU-intensive application is running in the 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.*

We should create 2 processes, one for performing the input and another for the output. Since we are using the one-to-one threading model, we are matching each thread we make to a kernel thread which allows the threads to run concurrently. Therefore, one thread can be used to read the contents of the file while the other can output the results or wait for more input. Any additional threads we create would be useless if input and output is all the program does.

b. *How many threads will you create for the CPU-intensive portion of the application? Explain.*

Since we have four processors available to us, we should create four threads for this portion (one thread for each processor). We will have each thread mapped to a kernel thread which will allow the program to run fully concurrently. If we create any less threads, then we will waste this potential concurrency, not maximizing the program's performance.

2. OSC 4.18: *Consider a multicore system and a multithreaded program written using the many-to-many threaded 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.*

a. *The number of kernel threads allocated to the program is less than the number of processing cores.*

This means each of the kernel threads will be mapped to a processer to run, but since we have more processing cores than kernel threads available, some of the processors will need to wait for a free kernel thread before they can run. This means that the program may run slower.

b. *The number of kernel threads allocated to the program is equal to the number of processing cores.*

Each kernel thread will get mapped to each processor, so all processors can run concurrently. However, since we have no spare kernel threads, some processes may need to wait if blocked.

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.*

Each of the processors will get mapped to a kernel thread so they may run concurrently with some kernel threads left to spare. If one of the processes gets blocked, one of the spare kernel thread may take over, which should help speed up the program.

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?*

IO-bound processes will typically have shorter CPU-bursts, therefore require less time with the processor. This is the reason why they will be favored by the algorithm. However, this will not permanently starve CPU-bound processes. This is because the scheduler will continuously run the IO-bound processes until they complete, or the amount of time used by the processor is greater than that of the CPU-bound processes. At this point, the CPU-bound processes should be favored by the scheduler, and should get a chance to run.

4. OSC 6.11: *Discuss how the following pairs of scheduling criteria conflict in certain settings:*

a. *CPU utilization and response time*

If we want to maximize CPU utilization, we want the CPU to stay busy as much as possible. This however comes at the cost of an increase in response time; the time between the request made and the first response. The more we maximize CPU utilization, the less time the CPU will spend waiting, so it will take more time for the CPU to respond to the request.

b. *Average turnaround time and maximum waiting time*

If we want to minimize the average turnaround time, then we want to reduce the average amount of time it takes to execute our processes. However, this would reduce the average waiting times of our processes; the time they spend in the ready queue. Therefore, this reduces the maximum waiting time.

c. *I/O device utilization and CPU utilization*

Utilizing I/O devices would mean increasing the amount of I/O bursts we perform, while CPU utilization would mean keeping the CPU as busy as possible. We cannot utilize both at the same time as there must be a balance between the number of I/O bursts and the number of CPU bursts.

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.*

The RM algorithm is a scheduling algorithm that schedules periodic tasks to run giving each task a static priority policy with preemption. The priorities are assigned based on the inverse of the task's period, meaning that tasks with shorter periods have a lower priority. Tasks with lower priorities will be preempted if other tasks with higher priorities become available. This

algorithm is classified as a fixed-priority algorithm because it assigns priorities to all the tasks it is asked to schedule and run, and those priorities do not change.

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?*

First, we find the schedulable utilization from the following

$V_{RM}(3) = 3 * (2^{1/3} - 1) = 3 * (1.259921 - 1) = 3 * (0.259921) = 0.779763$

Now we need to find the utilization for each T_i by computing the following

$U_1 = (1 / 4) = 0.25$; $U_2 = (2 / 8) = 0.25$; $U_3 = (2 / 20) = 0.1$

The schedule is feasible if $U_1 + U_2 + U_3 \leq V_{RM}$

$(0.25 + 0.25 + 0.1) = 0.6$ which is indeed less than V_{RM}, therefore there exists 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?*

First we find the new schedulable utilization

$V_{RM}(4) = 4 * (2^{1/4} - 1) = 4 * (1.189207 - 1) = 4 * (0.189207) = 0.756828$

We also need the utilization for the new task $T_4 = (1 / 5) = 0.2$

Now we check the sums

$(0.25 + 0.25 + 0.1 + 0.2) = 0.8 \geq 0.756828$

Therefore, the colleague is incorrect, as the value of 5 will not work. This is because the total utilization of the tasks given is strictly greater than the schedulable utilization.