Homework 3

ComS 353 - Gavin Monroe - gmonroe

1. (10 points) Ex 4.10 Which of the following components of program state are shared across threads in a multithreaded process? a. Register Values, b. Heap Memory, c. Global Variables and d. Stack Memory.

The threads of a multithreaded process share heap memory(b) and global variables(c). Each thread has its separate set of register values and a separate stack.

2. (10 points) Ex 4.11 Can a multithreaded solution using multiple user-level threads achieve better performance on a multiprocessor system than on a single processor system? Explain. Assume the many-to-one multithreading model is used.

No, in light of the fact that by and large, the client level strings are escaped the portion and as the working framework see's the multi-strung application as a solitary procedure to it won't appoint the strings to various processors that are accessible for use. In this way, as we can't plan various strings of the procedure to various processors subsequently no exhibition gain is accomplished.

3. (10 points) Ex 4.13 Is it possible to have concurrency but not parallelism? Explain.

It is possible to have concurrency but not parallelism because it is not possible to have parallelism without concurrency. For instance, consider a solitary center processor running two strings. A working framework will ordinarily switch to and fro between the two strings rapidly, giving the presence of parallelism. the two strings are each alternating executing their individual directions on a similar cpu center. Despite the fact that it is preposterous to expect to have parallelism without simultaneousness, it is conceivable to have simultaneousness however not parallelism.

4. (10 points) Ex 4.14 Using Amdahl's Law, calculate the speedup gain for the following applications:
• 40% parallel with (a) 8 processing cores and (b) 16 processing cores
• 90% parallel with (a) 4 processing cores and (b) 8 processing cores
Note: you do not need to answer the second part of the question stated in the textbook

Amdahl's law: speedup = 1 / ((1-p) + p/s) where as, p = fraction of code that can be parallelized, s = number of cores. 1: 40% parallel
a) 8 processing cores, p = 0.4, 1-p = 1 - 0.4 = 0.6, s = 8, speedup = 1 / (0.6 + 0.4/8) = 1.53
b) 16 processing cores p = 0.4, 1-p = 1 - 0.4 = 0.6, s = 16, speedup = 1 / (0.6 + 0.4/16) = 1.6

2: 90% parallel
a) 4 processing cores, p = 0.9, 1-p = 1 - 0.9 = 0.1, s = 4, speedup = 1 / (0.1 + 0.9/4) = 3.076
b) 8 processing cores, p = 0.9, 1-p = 1 - 0.9 = 0.1,s = 8, speedup = 1 / (0.1 + 0.9/8) = 4.70

5. (10 points) Ex 4.16 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 startup 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).
• How many threads will you create to perform the input and output? Explain.
• How many threads will you create for the CPU-intensive portion of the application? Explain

Think about the accompanying framework information:
• 2 Dual-center processors
• 4 booking processors
• Input and yield tasks
• CPU-concentrated application and CPU security

Just one (single) string to make for plays out the info and yield tasks.
Reasons are as per the following:
• Multithreading is utilized for simultaneousness of information and better execution. Strings are utilized to perform tasks equal.
•The upsides of strings are lessen time, improve CPU time, and keep away from more memory.
• Number of strings estimation relies on the booking need and needs of the application. Here doesn't specify values for information and yield tasks. In this way, accept all has the same qualities. At that point just a single string is sufficient for this sort of use. The purpose for checking one is that after that input activity and yield will come. Along these lines, there is no compelling reason to call different strings. In this way, making a solitary string is sufficient to play out the information and yield.

Make either 4 or 8 strings for the CPU-serious bit of the application.
• A framework has two-serious bit. It relies on the activities of planning processors and utilization of CPU. In this way, if the two activities are put away independently than 8 in any case 4 strings are sufficient. It is the equal methodology of the multithread. Thus, either 4 or 8 strings are sufficient for the CPU-concentrated segment of the application.

6. (10 points) Ex 4.19 The program shown in Figure 4.23 uses the Pthreads API. What would be the output from the program at LINE C and LINE P?

Child = 5, Parent = 0

The parent forks a child procedure which has its own memory space, at that point sits tight for the forked kid procedure to finish. The child procedure makes another string with a similar memory space as the kid procedure, and changes the estimation of the worldwide variable to 5. At that point it yields the estimation of that variable which is 5 and the child procedure wraps up. In the parent procedure, the estimation of the worldwide variable stays unaltered and its worth is yielded, which is 0.