Stamatios Morellas
COM S 352 - Homework 3
Due 2/12/20 @ 11:59pm

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
d. Stack memory

The **heap memory** is always shared across all threads in a multithreaded process. Answer b) is correct.

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.

The short answer is that **it depends**. If a system has dedicates one processor/core per user thread, which means that each thread may run in parallel, then a multiprocessor system will undoubtedly increase performance. In most other instances, using a single processor system (a single kernel thread) to process multiple user-level threads will satisfy performance requirements.

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

The answer is **yes**, it is possible to have concurrency but not parallelism. Parallelism is when two or more tasks begin their execution at the same time and work simultaneously, while concurrency is when two different tasks or threads are work together in an overlapping time period, but it does not necessarily mean that they have to run at the same moment. The nature of parallelism is that you are taking a large problem and dividing it up into smaller components, which can then be solved concurrently. Sometimes both parallelism and concurrency are required to complete a task and other times, only one of the two is required. Prioritizing the methods depends on the requirements of the operating system to select one over the other.

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
  - (b) 16 processing cores
- 90% parallel with
  - (a) 4 processing cores
  - (b) 8 processing cores

Note: you do not need to answer the second part of the question stated in the textbook.

Amdahl's law is given by the following equation:

$$S_{latency}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

Where:

$S_{latency}$ is equal to the theoretical speedup of the execution of the whole task

s is the speedup of the part of the task that benefits from improved system resources;

p is the proportion of execution time that the part benefiting from improved resources originally occupied

40% parallel with 8 processing cores:

$p = 0.4$

$1 - p = 1 - 0.4 = 0.6$

$s = 8$

$$speedup = \frac{1}{0.6 + \frac{0.4}{8}} = 1.54$$

40% parallel with 16 processing cores:

$p = 0.4$

$1 - p = 1 - 0.4 = 0.6$

$s = 16$

$$speedup = \frac{1}{0.6 + \frac{0.4}{16}} = 1.60$$

90% parallel with 4 processing cores:

$p = 0.9$

$1 - p = 1 - 0.9 = 0.1$

$s = 4$

$speedup = \frac{1}{0.1 + \frac{0.9}{4}} = 3.07$

90% parallel with 8 processing cores:

$p = 0.9$

$1 - p = 1 - 0.9 = 0.1$

$s = 8$

$speedup = \frac{1}{0.1 + \frac{0.9}{8}} = 4.70$

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.

System data:

- 2 dual-core processors (4 cores total)
- 4 scheduling processors
- I/O operations
- One-to-one threading (each user thread maps to a kernel thread)

In order to perform the input and output operations, I will use a **single thread**. Based on the requirements of this application, a single thread should be enough to perform these operations because the output operation is the successor for the input op, so there is no need to call any other threads, which means that it's sufficient to only use one.

For the CPU-intensive portion of the application, I will use **multiple threads**. Depending on the scheduling of operations as well as CPU usage, it may be more beneficial to use a higher number of threads, but since this system uses one-to-one threading, we should use a total of **4 threads**, one for each processor core.

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?

Output of line C:

```
CHILD: value = 5
```

Output of line P:

```
PARENT: value = 0
```