180 C Homework 3

Name:D Venkata Sai Sri Hari

SJSU ID-014533571

Reference- Silberschatz textbook, Internet

4.9 Can a multithreaded solution using multiple user-level threads achieve better performance on a multiprocessor system than on a singleprocessor system? Explain.

A.
No the kernel would not be aware of the user level threads that are created . Therefore it is not able to run the user level threads on different processors.

4.13 Determine if the following problems exhibit task or data parallelism:
- **The multithreaded statistical program described in Exercise 4.21**
- **The multithreaded Sudoku validator described in Project 1 in this chapter**
- **The multithreaded sorting program described in Project 2 in this chapter**
- **The multithreaded web server described in Section 4.1**

A.
• Task parallelism. Each thread is performing a different task on the same set of data.
• Task parallelism. Each thread is performing a different task on the same data.
• Data parallelism. Each thread is performing the same task on different subsets of data.
• Task parallelism.

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 startup and termination, the program is entirely CPUbound. 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.**

A.

1. 2 Threads as one for opening and one for closing. Creating additional threads provides no benefit
2. Four. There should be as many threads as there are processing cores.

4.15 Consider the following code segment:

```
pid t pid;
pid = fork();
if (pid == 0) { /* child process */
fork();
thread create( . . .);
}
fork();
```

a. How many unique processes are created?
b. How many unique threads are created?

A.
Let there be one parent process P. For first fork one child p1 is created. In the if statement p2 is created from p1 as only child has fork applied. For the next for P3,p4,p5, are created for p,p1,p2.

So total **6** process are created.

2 Threads are created.

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.
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 less than the number of user-level threads.

A.

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 all of the processors might be utilized simultaneously. However, problems arise when there are kernel thread blocks inside the kernel making the corresponding processor to remain idle.

When there are more kernel threads than processors, a blocked kernel thread could be used in favor of another kernel thread that is ready to execute, thereby increasing the utilization of the multiprocessor system.