Zoe Kosmicki

Assignment 4

Task 3

* What is race condition?

The race condition refers to how coded systems behave when dependencies exist in unpredictable elements.

* Why race condition is difficult to reproduce and debug?

Because they arise due to relative, interdependent interactions, errors that are caused by race hazards can become nonproblems when examined in a more controlled space, like a debugger.

* How can it be fixed? Provide an example from your Project A3

Looking at spmd2.c, the program we wrote for assignment 2, at first the output wasn’t functioning correctly because of how the Raspberry PI’s memory is organized. This was fixed by making the variable declarations private, so each thread would be executed in parallel without sharing due to the shared memory. This shows that race conditions are fixed by writing smart and careful code, being sure to trace out your code to be sure there are no hazards.

* Summarize the Parallel Programming Patterns section in the “Introduction to Parallel Computing\_3.pdf” (two pages) in your own words (one paragraph, no more than 150 words).

Parallel code can be very difficult to write, and because of this, programmers have created parallel patterns, which are rough guidelines on how to structure parallel code so that it runs and creates the desired output. While different patterns function and are organized in different ways, they all use the same basic building blocks to achieve their end goals.

* In the section “Categorizing Patterns” in the “Introduction to Parallel Computing\_3.pdf” compare the following:
  + Collective synchronization (barrier) with Collective communication (reduction)

Barriers are roadblocks in code that stop processes from continuing until every thread/process has gotten to a specific point. Reduction is an operation where a process can be reorganized to execute in a more efficient way. They’re both forms of management and communication within parallel computing.

* + Master-worker with fork join

Master-worker and fork join are similar, but with master-worker, each “worker” thread or process works in parallel with every other “worker”, with the master process finishing after each worker has finished and their individual outcomes have been computed together. In fork join, there are “regions” of parallelism, where a process can start sequentially and branch off to do tasks in parallel, and then rejoin together to continue sequentially.

* Where can we find parallelism in programming?

In programs (within the code itself), in data (how data is stored and used), and in a machine’s resources (how these resources are used and allocated).

* What is dependency and what are its types (provide one example for each)?

A dependency is when code needs to rely on and within itself to produce an output. The types of dependencies are true dependencies, where a later operation depends on a previous operation (abc = 7; d = abc), anti-dependencies, where a variable that is used in an operation gets changed after the operation (a = 9; b = a - 3; a = 100), and output dependencies, where if the order of the instructions were changed the output would change as well (a = 5; b = a + 7; a = 2).

* When a statement is dependent and when it is independent (Provide two examples)?

A statement is dependent when it relies on an earlier or later statement for its outcome (b = 7; a = b). A statement is independent when it can execute on its own without needing to rely on other statements (a = 10 + 20; b = 17 + 9).

* When can two statements be executed in parallel?

When no dependencies exist between the statements and they can be in any order without affecting the outcome.

* How can dependency be removed?

Some dependencies can be removed by rewriting/editing code. However, not all dependencies can be removed.

* How do we compute dependency for the following two loops and what type/s of dependency?

By examining the IN and OUT sets of the loops and performing dependency analysis to see where dependencies exist in the code. In the first loop, there’s true dependency between the loop index and a. In the second loop, there’s true dependency between the loop index, a, and b.