Nathan Heckman

A4 Parallel Programming Foundations

**What is a race condition?**

A race condition is behavior that is dependent on the timing of uncontrollable events. They occur in logic circuits and multithreaded or distributed software programs.

**Why are race conditions difficult to reproduce and debug?**

The end result is nondeterministic and depends on thread timing, which can greatly vary. It’s difficult to get the exact same output between threads more than once.

**How can it be fixed? (A3 Example)**

Write software carefully and with race conditions in mind as race conditions are easier to debug while writing the program than afterwards. An example from A3 would be using Uniform Memory Access (UMA) to allow threads to share memories and synchronize, which would fix the issue of dependency and race conditions.

**Parallel Programming Patterns Summary:**

Developers can use parallel programming patterns to make reusable code that works well for different types of programs. Learning and writing good parallel code is important for new programmers and veterans alike. The two main organizational patterns developers use are strategies and concurrent execution mechanisms. Strategies can be broken down into algorithmic and implementation strategies which both work together to make an efficient program. Concurrent execution mechanisms include process/thread control and coordination, which can be broken down into message passing and mutual exclusion.

**Barrier and Reduction Comparison:**

Both are a type of concurrent execution mechanism and fall under the coordination classification.

**Master-Worker and Fork Join Comparison:**

Both are a type of implementation strategy under the category of program structure. Other strategies in this classification include single program, multiple data, and parallel for loop.

**Where can we find parallelism in programming?**

Parallelism is found at the program statement level as well as the process level. As programmers we must determine which statements can be run simultaneously and the computer must determine how to run processes simultaneously.

**What is dependency and what are its types?**

Dependency is when one operation relies on another to be completed for it to give the correct output. Types of dependences include:

True dependences – A=1, B=A

Anti-dependences – A=B, B=1

Output dependences – A=f(x), A=B

**When is a statement dependent and when is it independent?**

Statements are dependent when the order of their execution affects the outcome of the program. Statements are independent when their order of execution doesn’t matter to the end result of the program. If two variables are used in a single line/statement, they are usually dependent on one another.

**When can two statements be executed in parallel?**

Only when there are no dependences between the two statements.

**How can dependency be removed?**

Some dependences can be removed by rearranging or eliminating statements from the program.

**How do we compute dependency for the following loops? What type?**

Left loop – executed sequentially and only a single statement. Each iteration after the first is dependent on the previous one to be executed in order for it to execute. True dependency

Right loop – executed parallel. There is no dependency between the statements since they can only be executed in parallel when there are none.