CSCI-UA.0480-001 Special Topic: Multicore Programming

Homework 1

Due June 13, 2019

Please solve the following and upload your solutions to your private GitHub repository for the class as homework1.pdf by 11:59pm on the due date above. If for some reason this poses a technical problem, or you wish to include diagrams that you don't wish to spend time drawing in a drawing application, you may hand in a printed copy (*not* hand-written) at the beginning of class (7:10pm) on the day of the deadline. **Unlike labs, late homeworks will be assigned a grade of 0.**

---

1. Explain the difference between concurrency and parallelism with an example: if an operating system is executing three long-running programs, how would its scheduler execute the programs concurrently on one core, concurrently on three cores, or in parallel on three cores? Comment on running the programs in parallel on one core.

2. You are implementing a server that sends responses to requests, and requires very small amounts of computation to handle each request. Your application requires many clients and many servers to work together. Which of the programming models discussed in class are you likely to use to model this application, and why?

3. Apply Amdahl's law to compute the speedup for the following program if you have (a) 1, (b) 2, (c) 4, (d) 8, (e) 12, (f) 16, and (g) ∞ CPUs. In the following diagram, **S** portions are sequential and **P** are parallelizable.

.S	P	S	P	P	S
15%	18%	12%	15%	30%	10%

4. What are hazards in a CPU pipeline? Why and how do they reduce the effective IPC (instructions per clock)? Feel free to explain whatever concepts of how pipelines work are relevant.

5. Draft the skeleton of a family of C++ classes (declarations, no definitions) that could be used to represent `struct` and `class` concepts in C++. For example, you might want an `Object` class that is the parent of the `Struct` and `Class` classes, which would contain *other things*. Think about storing the things that can go in either one (member variables, methods), and what we need to know (type or return type, visibility, arguments, etc.). An exhaustive representation of every relevant C++ is not necessary, but show you understand the concepts.