CSCE 435 Fall 2015

HW 3: Game of Drones

Due date: 11:59pm Wednesday, October 7, 2015

The goal of this assignment is to develop a multithreaded code to find the location of a drone in a two-dimensional grid in the shortest time possible.

You are provided with the program `drone.c` that has serial code to initialize a grid with a drone in an unknown location and to determine the location of the drone. You are expected to replace the serial code in the file with multithreaded code to find the location of the drone. The goal of your parallel implementation should be to minimize the execution time.

A header file `drone.h` is also provided that has data structures and routines associated with the grid. A summary of the important routines provided in the header file is given below.

- `initialize_grid`: Initialize the grid size and place the drone in a random location; initialize the duration of sleep after each call to check_grid;
- `check_grid(i,j)`: check if the drone is present at grid location `(i,j)`, return 0 if the drone is at `(i,j)`, otherwise return an integer k such that the drone is within k steps of `(i,j)` (a step equals the distance between two adjacent nodes of the grid); note that k is the smallest multiple of 16;
- `check_drone_location(i,j)`: check if the location you determined is correct (to be used to determine if the computed solution is correct).

The header file should not be modified in any way. You are to assume that you cannot access the drone location `(_drone_i,_drone_j)` from your code directly, and must use `check_grid()` to search for the drone.

1. (60 points) Develop a multithreaded implementation to determine the location of the drone using the interface routines provided in the header file.
2. (20 points) Describe your parallel algorithm and estimate its performance on p processors for an n×n grid. Assume that each call to `check_grid` requires Q seconds. Also assume that calls to this routine can be made concurrently.
3. (20 points) Your code should compute the solution quickly and efficiently, utilizing all available processors on a single compute node of `ada`. Points will be assigned based on the average total time taken by your code as compared to the rest of the class. Testing will include problems with a variety of grid sizes. The sleep time in `check_grid` may also be varied to test how the code performs.

Submission: You need to upload the following to ecampus:

1. Problem 1: submit the file `drone.c`; you do not need to submit `drone.h`
2. Problem 2: submit a single PDF or MSWord document with your.