CSCI 3366 (Introduction to Parallel and Distributed Processing), Fall 2005

Homework 4

Assigned: November 18, 2005.

Due: November 30, 2005, at 5pm.

Credit: 30 points.

> *Note:* The HTML version of this document may contain hyperlinks. In this version, hyperlinks are represented by showing both the link text, formatted like this, and the full URL as a footnote.

1 Overview

Your mission for this assignment is write a parallel version of mathematician John Conway's "Game of Life", as described in class November 9. The Game of Life is not so much a game in the usual sense as a set of rules for a cellular automaton — there are no players, and once the initial configuration is established, everything that happens is determined by the game's rules. The game is "played" on a rectangular grid of cells. Some cells are "live" (contain a simulated organism); others are "dead" (empty). At each time step, a new configuration is computed from the old configuration according to the following rules:

- For each cell, we look at its eight neighbors (top, bottom, left, right, and the four diagonal neighbors) and count the number of cells that are live. (Notice that this count is based on the configuration at the start of the time step.)
- A dead cell with exactly three live neighbors becomes live; otherwise it stays dead.
- A live cell with two or three live neighbors stays live; otherwise it becomes dead (of isolation or overcrowding).

2 Details

2.1 Sequential starter program

To help you get started, I wrote a sequential C program with a simple text interface. Here it is, with the part of the program that actually does the calculations removed (but all the tedious stuff to get input and print output left in):

- Code: game-of-life.c[1].
- Sample input file: in1.txt[2].

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS3366_2005fall/Homeworks/HW04/Problems/game-of-life.c`

[2] `http://www.cs.trinity.edu/~bmassing/Classes/CS3366_2005fall/Homeworks/HW04/Problems/in1.txt`

You don't have to use this code, but it does illustrate one approach to dealing with 2D arrays in C (simulating them with 1D arrays). You might find it useful to start by filling in the parts of the code I left out and running the result a few times, to test that you understand how to do the computational part of the game. Or you might choose to start from scratch. If you do, and you choose a different user interface, be sure to include comments in/with your program that explain how to run your program.

2.2 Parallel programs

Your mission is to write two parallel versions of this application, one for distributed memory (e.g., using MPI) and one for shared memory (e.g., using OpenMP — though you could rewrite in Java if you prefer). You don't have to include code to time your program, but it might be interesting to do so.

3 What to turn in and how

Submit your program source code by sending mail to `bmassing@cs.trinity.edu`, with each file as an attachment. Please use a subject line that mentions the course number and the assignment (e.g., "csci 3366 homework 4"). You can develop your programs on any system that provides the needed functionality, but I will test them on one of the department's Fedora Core 4 Linux machines, so you should probably make sure they work in that environment before turning them in.