COSC 40703 - High Performance and Parallel Computing

Semester Project

Due: April 30^{th}, 2024

Objective

The purpose of this semester programming project is to reinforce your skills in developing parallel programs. This can be accomplished by analyzing one of the problems below, programming a solution, capturing and analyzing the output, and presenting your findings (written and orally) to the class.

Getting Started

For this programming project, you are use TCU's supercomputer or server that uses GPGPUs. You may use any of the parallel programming paradigms presented in class or in the textbook (OpenMP, pthreads, C++ threadx, MPI, CUDA, etc.)

1 Assessment and Grading

This project must be done in teams of one or two. Your team is to write a well-documented, well-structured C/C++ parallel program using procedures and functions. Programs that are all in the "main function" or have excessively long functions will result in loss of points. Your source code must use procedures and functions and be documented. You may use other parallel programming libraries and graphics engines in your solution.

You will need to verify that your program is generating correct output. Furthermore, you will need to run your program several times and measure program's performance by varying the input size, input parameters, number of processors, or a combination of all three.

You will need to capture the output of this program in some fashion as to present your results graphically or by use of an animation.

Your team will also write a report of no less than 8 pages (two-column format) suitable for submission to a computing conference. The exact template will be sent to you via email in the next few weeks.

On our final exam day, your team will give a 25 minute presentation (20 minutes presentation + 5 minutes Q/A) of your project. You will discuss the problem, approach and design methodologies, and algorithm to solve your problem. You will also give a demonstration to the audience of your

program in execution (live). You will also present your results and analysis of your work.

This project is worth 300 points.

2 Submitting Your Project

Create a subdirectory/folder with your team name. In this folder, put the following in files:

- Your LaTeX file of your results and discussions from the two parts of this project. Be sure to also include any PDFs or figures that your LaTeX file may need to include.
- A 'Makefile' that is capable of building both of your executables (by just typing 'make' at the command line).
- The source code for your programs.
- Charts, graphs, spreadsheets, animations, etc. should also be included in this folder.

Do not submit any auxiliary LaTeX files, object files, or executable files. I will build your LaTeX file into your report PDF. Zip your subdirectory/folder and submit it to the dropbox on TCUOnline.

Project A: Rabbit-Fox Island

Once upon a time there was an island populated only by rabbits, foxes, and vegetation. The island (conveniently enough) was the exact shape of a chessboard. Some local geographers have even drawn gridlines the serve to divide the island into 64 squares to facilitate their demographic studies on the populations of each inhabitant.

Within each square the populations of rabbits and foxes are governed by several factors:

- the population of rabbits and foxes in each square at the start of this "day"
- the reproduction rates of rabbits and foxes (the same over the entire island) during this day
- the vegetation growth rate
- the death rates of "old" rabbits and foxes during this day
- the eating habits of foxes (foxes live entirely on rabbits; when the vegetation is dense, rabbits are more difficult to find)
- the eating habits of rabbits (they live on vegetation; too many rabbits in a square could lead to their starvation and/or a lower reproduction rate and/or being easier for foxes to find and eat)
- the migration (from day to day) of rabbits from one square to other squares that are immediately adjacent
- the migration of foxes from one square to any other square within two "leaps".

Since this is an island, there are certain boundary conditions: The 28 squares on the water's edge have no migration possible into or out of the island for either rabbits or foxes. Similarly, there are certain initial conditions representing the starting populations of rabbits and foxes in the various squares at the time your program begins execution.

Your job is to simulate 100 (or 1000) years of life on the island, using time steps of a day in length, and to determine the populations of rabbits and foxes at the end of the period in each square on the island. For each pair of rabbits in a square at the start of a birthing day, which occurs every nine weeks, a litter of babies is born. The size of the litter ranges between two and nine and varies based on both food supply (vegetation level) and the number of rabbits in that square at the start of the day (population density), as given in Table 1. For each pair of foxes in a square at the start of a birthing day, which occurs every six months, a litter of kits is born. The size of the litter ranges between zero and five and varies based on both the food supply (rabbit population) and the number of foxes in that square at the start of the day (population density), as given in Table 2.

A fox can survive on as little as two rabbits per week, but will eat as many as four if they can be found. If the vegetation level is below 0.6, rabbits are more easily found. In that case, on any given day, there is a four in seven chance that a fox will eat a rabbit if there are sufficient rabbits available; if there are fewer rabbits than that, or if the vegetation level is at or above 0.6, the foxes will have to make do with a two in seven chance of having a meal - provided there are sufficient rabbits available at that consumption level. (If there are fewer rabbits than the number needed to

keep the fox population alive, foxes that did not get fed have a 10 percent chance that they will die off; that is, in addition to their natural death rate.) The lifespan of a fox is estimated to be four years. Use a random-number generator each day to determine whether one or more foxes die a natural death.

Each rabbit consumes vegetation; each rabbit consumes 0.1 percent of the vegetation in a square per day, under non-food-constrained situations. The normal lifespan of a rabbit is estimated to be 18 months. If the vegetation level is less than 0.35, the death rate due to starvation rise dramatically, as given in Table 3.

Use a random-number generator each day to determine the number of rabbits that dies from a combination of starvation and natural causes. The vegetation level rises quite rapidly when not being eaten by rabbits; growing conditions are ideal on the semitropical island. The vegetation level follows the growth/consumption formula:

$$\begin{aligned}\text{Vegetation at end of day} =& (110\%\text{of vegetation at start of day}) \\ & - (0.001 \times \text{number of rabbits at start of day})\end{aligned} \tag{1}$$

within the limits that the vegetation level will not drop below 0.1 or grow to more than 1.0. At the end of each day, 20 percent of the rabbit population randomly emigrates to adjoining squares. Use a random-number generator to determine the number that actually emigrate to each of the possible adjoining squares. Similarly since foxes range more widely, at the end of each day, every fox randomly emigrates zero, one or two squares distant from its location a the start of the day. Note: All possible migrations are to be considered uniformly likely among the choices available.

Case 1: Uniformly, there are two foxes and 100 rabbits per square initially; the vegetation level is 1.0 everywhere.

Case 2: There are 20 foxes in one corner square and none elsewhere; there are 10 rabbits in every square except in the corner square diagonally opposte the foxes, and it contains 800 rabbits; the vegetation level is 0.3 everywhere.

Case 3: There are no foxes on the island, but there are two rabbits in each square; the initial vegetation level is 0.5 everywhere.

Vegetation at start of day	Number of rabbits at start of day				
	<2	2 to 200	201 to 700	701 to 5000	>5000
<0.2	0	3	3	2	2
>= 0.2 but <0.5	0	4	4	3	3
>= 0.5 but <0.8	0	6	5	4	4
>= 0.8	0	9	8	7	5

Table 1: Rabbit Births

Rabbit population at start of day	Number of foxes at start of day				
	<2	2 to 10	11 to 50	51 to 100	>100
<3.0	0	2	2	1	0
>= 3.0 but <10	0	3	3	2	1
>= 10 but <40	0	4	3	3	2
>= 40	0	5	4	3	3

Table 2: Rabbit and Fox Populations

Vegetation Level	Rabbit Lifespan
0.1 to 0.15	3 months
0.15 to 0.25	6 months
0.25 to 0.3	12 months
over 0.35	18 months

Table 3: Rabbit Lifespan

Project B - The Softie

An enterprising entrepreneur has concluded that the ideal opportunity for him to combine his love of dogs with a vast fortune is to corner the market for a new breed of dog: the Softie.

Characteristics of the Softie are as follows:

A. **Length of Coat**: 8 in. or longer
B. **Coat Characteristics**: Extremely soft (or softer), brown areas on white background, white paws, black tail
C. **Tail Characteristics**: Short (4 in. to 6 in.), points straight up
D. **Weight Extremely large**: 90 kg or heavier
E. **Foot Characteristics**: Paw print area in excess of 9 sq. in., fully webbed between toes
F. **Disposition**: Extremely mild tempered

Each of these characteristics has a range when viewed across all dogs:

A. Can be represented by eight bits in which 00000000_2 corresponds to a hairless dog, 11111111_2 corresponds to a hair length of 10.2 in., and 11001000_2 corresponds to 8 in.
B. Can be represented by six softness bits (in which 000000_2 corresponds to the ultimate in softness), 000111_2 is extremely soft, and 111111_2 is the ultimate in stiffness; three background brightness bits, in which 000_2 is bright and 111_2 is dark; three background color bits, in which white is 000_2 and brown is 001_2 and all other colors are covered by the remaining size combinations; three foreground brightness bits (000_2 is bright); three foreground color bits (000_2 is white, 001_2 is brown, 010_2 is red, 011_2 is yellow, 111_2 is black, and the remaining bit patterns are other colors); one bit for paw color, in which 0 corresponds to white and 1 is "anything else"; one bit for tail color, in which 0 corresponds to "anything else" and 1 corresponds to black.
C. Can be represented by 10 bits, in which eight are related to tail length (00000000_2 corresponds to a tailless variety, and 11111111_2 corresponds to a tail-length of 25.5 in. or longer) and two denote tail appearance: 00_2 correspond to pointing straight up, 01_2 correspond to pointing horizontally, 10_2 corresponds to pointing straight down, and 11_2 corresponds to the highly undesirable curly-tailed appearance.
D. Can be represented by 10 bits, in which the first seven correspond to a weight in kg and the second three correspond to a weight in increments of $\frac{1}{8}$ kg. Thus, a weight characteristic of 00000101011_2 would be a weight of $5\frac{3}{8}$ kg.
E. Can be represented by 10 bits, seven bits of which correspond to pawprint area and three to the fraction of webbed. In this example, 000000_2 corresponds to a pawprint area of 0.5 sq. in., 1111111_2 corresponds to a pawprint area of 13.2 sq. in., and 0100011_2 corresponds to a pawprint area of 4.0 sq. in. In the final three bits, 000_2 corresponds to 1/8 webbed, and 111_2 corresponds to fully webbed.
F. Can be represented by six bits, in which 000000 corresponds to the ultimate in mild-tempered-disposition, 000100_2 is extremly mild tempered, and 111111_2 is "meaner than a junkyard dog" - the ultimate in non-mild-temperedness.

Thus, in total, one needs 64 bits to represent an individual dog in this population. One such dog might be

11001000 001011 000000 111001 01 1000000011 001101100 0100011 011 00100

whose characteristics are:

- $[11001000_2]$ 8 in. coat
- $[001011_2]$ Slightly more stiff than desirable
- $[000000_2]$ Background color is bright white
- $[111001_2]$ Foreground color is dark brown
- $[01_2]$ White paws and black tail
- $[1000000011_2]$ 12.8 in. long, curly-tailed
- $[001101100_2]$ 13.5 kg weight
- $[0100011_2]$ 4 sq. in. pawprint area
- $[011_2]$ 50 percent webbed
- $[000100_2]$ Extremely mild tempered

In short, this dog has some desirable characteristics (its coat is pretty good, its paw size is find, and its temperament is fine) and some that are undesirable (coat is a little stiffer than the standard, the tail is curly, the dog is extremely small compared to the standard, and it will not be a good water dog as desired due to its less than fully webbed toes).

The entrepreneur is anxious to hedge his bets on this project and hired both you and a competitor to develop Softies. Your competitor has a single large kennel complex in which he plans to house 500 dogs; you have opted for five independent kennel locations each housing 100 dogs. Both of you plan to visit animal shelters and take the first 500 nonspayed/neutered dots you find (i.e. you will start off the initial population with "random" characteristics).

Bot your competitor and you plan to keep the population sizes constant in your kennels while breeding successive generations of dogs. You both plan to have each kennel location two of its parents in the current generation that best match the standard for the Softie breed. In addition, since a pair of parents may have multiple offspring in a given litter (corresponding to making multiple "cuts" and crossovers), both of you plan to give away (to your friends) the offspring that least conform to the standards for Softies, as needed, to keep your local populations constant.

The major difference between the two approaches you and your competitor are using is that he has a single large population of while you have five independent kennel "subcontractors" each working with an isolated subpopulation.. He will simply produce n generations of offspring; your subcontractors each will produce five generations of offspring in isolation, and only then send their two most fit dogs to their neighbors (A sends one to B and one to E, B sends one to A and one to C, etc.), as in the island model.

1. Write a single-processor genetic algorithm solution modeled after your competitor's approach. Run it with several random initial populations and compute the average number of generations it takes before 10 percent of the population (50 dogs) meets the standards for Softies.
2. Write a multiprocessor/multicomputer genetic algorithm solution modeled after your approach. As in in Part 1, run this with several random initial populations and compute the average number of generations it takes before a total of 50 dogs from your kennels meets the standards for Softies.

Hint: You will need to construct an evaluation function that incorporates all of these characteristics, such that a dog meeting the characteristics of the Softie standard will exhibit a greater fitness than one that does not.

Project C - Sharks and Fishes

Another simple fun example of cellular automata is "Sharks and Fishes" in the sea, each with different behavioral rules. An ocean could be modeled as a three-dimensional array of cells. Each cell can hold one fish or one shark (but not both). Fish might move around according to these rules:

1. If there is one empty adjacent cell, the fish moves to this cell.
2. If there is more than one empty adjacent cell, the fish moves to one cell chosen at random.
3. If there are no empty adjacent cells, the fish stays where it is.
4. If the fish moves and has reached its breeding age, it gives birth to a baby fish, is left in the vacated cell.
5. Fish die after x generations.

The sharks are governed by the following rules:

1. If one adjacent cell is occupied by a fish, the shark moves to this cell and eats the fish.
2. If more than one adjacent cell is occupied by the fish, the shark chooses one fish at random, moves to the cell occupied by the fish, and eats the fish.
3. If there are no fish in adjacent cells, the shark chooses an unoccupied adjacent cell to move to in the same way that fish move.
4. If the shark moves and has reached its breeding age, it gives birth to a baby shark, which is left in the vacated cell.
5. If a shark has not eaten for y generations, it dies.
6. Sharks die after z generations. ($y < z$ and $x < z$)

Write parallel programs to simulate the actions of the sharks and fish as described above. The parameters that are input are the size of the ocean, number of fish, number of sharks, their initial placement in the ocean, breeding ages, and shark starvation time. Adjacent cells do not include diagonally adjacent cells. Therefore, there are six adjacent cells, except for the edges. For every generation, the fishes' and sharks' ages are incremented by 1. Your simulation should also take into account currents in the water. Suggested tools are MPI, OpenMP, and CUDA (pick two).

Project D - The Intersection

Simulate a road intersection controlled by traffic lights. The simplest intersection is a four-way with two-lane roads. Vehicles come from all four directions along the roads and wish either to pass straight through the junction to the other side, or turn left, or turn right. On average, 70 percent of the vehicles wish to pass straight through, 10 percent wish to turn right, and 20 percent wish to turn left. Each vehicle moves at the same speed up to the junction. Develop a set of driving rules to solve this problem by a cellular automata approach, and implement them in parallel programs using your own test data (vehicle numbers and positions). Suggested tools are MPI, OpenMP, and CUDA (pick two).

3 Project E - Crossy Log Game

Write a multithreaded program to implement the following arcade game: A river has logs floating downstream and upstream. A frog must cross the river by jumping on logs as they pass by. The user controls when the frog jumps, which can only be perpendicular to the riverbanks. You win if the frog makes it to the opposite side of the river, and you lose if the frog lands in the river. Graphical output is necessary, and sound effects are preferable. Concurrent movements of the logs are to be controlled by separate threads. Suggested tools are OpenMP or CUDA.

4 Project F - Image Morphing

You have been commissioned by a major file studio to develop a really fast "morphing" package that will change one image into another. You come up with the idea of having two images, the original image and the final image, and changing each pixel on the original image to become closer and closer to the pixels of the final image in a lock-step SIME fashion. This method is certainly embarrassingly parallel, although it may not create a very smoothly changing shape. Experiment with the method by writing parallel programs and demonstrate it to the studio using pictures of actors (or your friends...but not your instructor). Suggested tools are OpenMP, CUDA, and MPI (pick two).