Wator Project

Dr. Joseph Kehoe

November 3, 2023

0.1 Overview

You are required to implement the “Wa-Tor” simulation. The full specification of this simulation is provided in the original Mathematical Recreations column.

A.K. Dewdney: ”Computer Recreations; Sharks and Fish wage an ecological war on the toroidal planet of Wa-Tor” Scientific American. pp. I4—22.

Online descriptions for this simulation can be found at:

- `https://en.wikipedia.org/wiki/Wa-Tor`

0.2 Implementation

The project must be implemented on Linux using the gcc compiler and documented with Doxygen. This simulation must use openMP for parallelization. If you wish to run this on a GPU then either CUDA or openMP may be used. The simulation must show a simple graphical output representing the population at each step.

0.3 Parameters

The simulation will take seven parameters when run:

NumShark Starting population of sharks;

NumFish Starting population of fish;

FishBreed Number of time units that pass before a fish can reproduce;

SharkBreed Number of time units that must pass before a shark can reproduce;

Starve Period of time a shark can go without food before dying;

GridSize Dimensions of world;

Threads Number of threads to use.

0.4 Rules

The full set of rules are available online and are copied here for your convenience.

Time passes in discrete jumps, which I shall call chronons. During each chronon a fish or shark may move north, east, south or west to an adjacent point, provided the point is not already occupied by a member of its own species. A random-number generator makes the actual choice. For a fish the choice is simple: select one unoccupied adjacent point at random and move there. If all four adjacent points are occupied, the fish does not move. Since hunting for fish takes priority over mere movement, the rules for a shark are more complicated:

from the adjacent points occupied by fish, select one at random, move there and devour the fish. If no fish are in the neighborhood, the shark moves just as a fish does, avoiding its fellow sharks.

0.4.1 Fish

- At each chronon, a fish moves randomly to one of the adjacent unoccupied squares. If there are no free squares, no movement takes place.
- Once a fish has survived a certain number of chronons it may reproduce. This is done as it moves to a neighbouring square, leaving behind a new fish in its old position. Its reproduction time is also reset to zero.

0.4.2 Sharks

- At each chronon, a shark moves randomly to an adjacent square occupied by a fish. If there is none, the shark moves to a random adjacent unoccupied square. If there are no free squares, no movement takes place.
- At each chronon, each shark is deprived of a unit of energy.
- Upon reaching zero energy, a shark dies.
- If a shark moves to a square occupied by a fish, it eats the fish and earns a certain amount of energy.
- Once a shark has survived a certain number of chronons it may reproduce in exactly the same way as the fish.

0.5 Deliverables

Deliverables are as follows:

Code Must be on github with link and permissions supplied to me;

Results Tables and graphs showing speedups for simulation when run with one, two, four and eight threads. These must also be on github.

0.6 Due Date

The project is due on 31st November at 5pm.