

Master 1 – Parallel Algorithms and Architectures Homework

Benjamin DEPARDON

(based on an idea of Loris MARCHAL and Cédric TEDESCHI)

Work to be realized in groups of 2 for November, 30th, 2009.

This homework consists in simulating an ocean within which sardines and sharks coexist. The ocean will be assimilated to a 2D toric grid. Initially, each square of the grid can contain a shark or a sardine. Each specie's population evolves with time (discrete) according to certain local rules.

Concerning sardines:

- At each step, a sardine tries to move to a nearby randomly chosen square. If this latter isn't empty, the sardine does not move. Sardines can move to the top, the bottom, the right or left, but cannot move diagonally.
- Once a given age attained (A_{sardine} , in number of steps) and when moving, sardines can procreate: when leaving a square, a sardine gives birth to a youngling (of age 0) with a probability p_{sardine} , and abandon it on the square it left. If a sardine do not move, it cannot procreate.
- Sardines do not die unless they are eaten by a shark.

Concerning sharks:

- Sharks move just like sardines do, they randomly choose a nearby square among the ones above, below, to the left or to the right of them. If this square already contains a shark, the move does not happen. If it is occupied by a sardine, then the shark moves and eats the sardine.
- Sharks reproduce the same way sardines do: if a shark reaches the reproduction age A_{shark} and moves, then it leaves a youngling (of age 0) behind itself with probability p_{shark} .
- Sharks can only eat sardines. If a shark do not eat for $T_{\text{starvation}}$ time steps, it dies.

This work is composed of two parts that have both a temporal and spatial dependency: you need to think deeply about the first part before starting the second part. When you'll describe your algorithms, you'll need to precise the data structures you use, and explain how they work. You have to send back your work for the above mentioned date: your code and a report (preferably in) for the following questions.

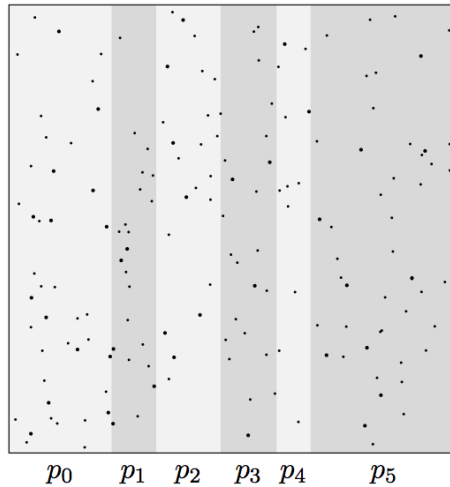


Figure 1: Initial heterogeneous status example: strip allocation on 6 processors, with 112 sharks and 134 sardines. (Bold points represent sharks, small ones represent sardines.)

1 Fish Origamis

In order to parallelize the simulation, we propose to divide the ocean into strips, as presented on Figure 1. Each strip is allocated to a processor, and there are as many strips as there are processors. The most important thing in a distributed simulation, is that the result is independent of the number of processors. In particular, the result has to be same as the one obtained with a sequential simulation on a single processor.

Question 1. *A priori, what in our simulation will be a problem in order to guaranty that the behavior in parallel is the same as in the sequential simulation? Thereafter, you'll take care to justify that your algorithms do not modify the behavior in parallel.*

Question 2. *Data needs to be exchanged between processors at the boundaries between strips of the ocean. Propose a protocol minimizing the number of exchanged data. Precise the conditions for your protocol to work, and the optimality cases.*

If the fish distribution is homogeneous, the strips allocated to the processors can have the same width. However, with a heterogeneous distribution containing dense areas, and others less dense, some processors will have more work, hence the global simulation will be slowed down.

For the time being, we consider that a processor knows the entire ocean, and that communications are costless.

Question 3. *Propose an optimal algorithm for cutting the ocean and balancing the load depending on the initial distribution of the fishes in the ocean¹. Do not forget that a processor has to update all fishes and process all squares in its ocean strip.*

We now suppose that communications are way more costly than computation (and of course we have at least two processors!)

¹Column 0 is not a mandatory frontier...

Question 4. *Propose an algorithm, for cutting the ocean, that minimizes the data exchanged between processors on as many steps as possible (in the mean). Make sure that at least each processor receives a fish if possible.*

We now suppose that we do not know anything *a priori* about the computation and communication speeds.

Question 5. *Propose a general algorithm that optimizes the time spent computing and communicating.*

In practice, we suppose that the ocean is too big to be kept in the memory of only one processor. Moreover, as fishes move, the initial configuration is modified, and what could have been an efficient cut, can become a poor one. The initial cutting is in fact chosen arbitrarily (with all strips having more or less the same width); the goal is then to modify the cutting of the ocean, whenever needed, in a more accurate way, using local information (as we cannot have on one processor the whole ocean).

Question 6. *Propose a mechanism to detect load imbalance in the current cutting. Propose a distributed algorithm for modifying the current distribution of the ocean. Clearly justify your newly chosen cutting, and the effective setting up of this new cutting.*

Question 7. *Estimate the quality loss due to your distributed cutting compared to the centralized one. Illustrate this by an example.*

We now suppose that we have $p = k^2$ processors, and that the ocean is cut using rectangles, not necessarily identical.

Question 8. *Explain the difference between this cutting and the cutting using strips: present the advantages or drawbacks in terms of performances for the simulation (computations and communications), and in terms of development efforts (complexity of the algorithms for communications, for load imbalance detection, and for modifying the cutting).*

2 MPI (Moving Plenty of sardInes)

The goal of this section is to write an MPI implementation of a distributed simulator. Your implementation will have to take as input a description file such as:

```
1000                (grid size : 1000 × 1000)
1 3 sardine 3       (fishes desctiption: x y type age)
3 5 shark 12
⋮
```

You can find on my webpage² two codes: `gen_homo.c` and `gen_hetero.c`, they respectively generate a homogeneous and a heterogeneous distribution of fishes. You will also find an example of sequential simulator `simul_seq.c`, that you can use to test different sets of parameters, and also take inspiration from.

We choose an allocation by vertical strips of size: $size \times size/p$.

²http://graal.ens-lyon.fr/~bdeparado/Educ_AlgPar_09.html

Question 9. Write an MPI code to initialize the simulation. Each processor can read the description file, but only keeps the information corresponding to its strip. Moreover, at each step, after updating its area, each processor will have to print its number and the number of sharks and sardines present in its area. Find a mechanism enabling the processors to print this information in the right order.

Question 10. Implement the whole simulation. For the time being, do not take into account the modification of the initial cutting. Base your implementation on the algorithms you gave in the first part for the communications (you can possibly implement simpler versions by justifying your choices).

Test your implementation on oceans of different sizes, with homogeneous and heterogeneous distributions. Compare the performances in the different cases, both in terms of computation (is the load well balanced?) and communications.

After a certain number of steps, the global fishes' positions may have changed, and hence the data distribution on the processors may not be optimal anymore. Or, generally speaking, the initial distribution is heterogeneous.

Question 11. Implement a mechanism for detecting load imbalance, and for creating a new cutting of the ocean based on your previous algorithms. Print the new cutting (the new boundaries for each processor) whenever a modification is made.

Compare the performances with the cutting with fixed strips.

Question 12 (Bonus). Implement a cutting based on squares of size: $size/k \times size/k$ on $p = k^2$ processors. (You do not need to worry about the modification of the initial cutting.)

Question 13 (Bonus). Create a processor dedicated to printing a minimal representation of the ocean: total number of sharks, total number of sardines, number of sharks and sardines per area, and size of the areas. The information mustn't be updated at each step, but rather periodically (100 steps, or 1 second, etc.) Use a graphical window with pixels of different colors if the ocean is small enough.

Also make the processor responsible of the parsing of the ocean description file, and of the data allocation to the other processors. You need to take into account that this processor cannot hold in memory the whole ocean.