|epcc|


# Parallel Design Patterns
# Assignment 1

B087928

February 21, 2016

# 1 Parallel Pattern and explanation

## 1.1

The parallelisation pattern selected for this problem is Geometric Decomposition.

## 1.2

The first step in deciding upon this parallelisation strategy was to find concurrency within the problem. We have 6 clear tasks that need to be done each iteration, the first 4 are specific to each squirrel: Calculate new position; (decision) does squirrel die? (decision) does squirrel reproduce? (decision) does squirrel catch disease? The last two of these have dependencies, and vice versa, on the next two tasks which are values pertaining to cells (geographical areas of the landscape): calculate populationInFlux and calculate infectionLevel.

The arithmetic operations required to complete these tasks are very simple once we have the relevant data readily available. Thus this problem is very data dependent. Therefore we shall benefit by using a design pattern with a good decomposition of data.

For each squirrel we must record the value of populationInFlux and infectionLevel for each step. Similarly for each cell we must know how many squirrels, and whether they are healthy or diseased, there are in that cell to calculate populationInFlux and infectionLevel at each step. This means that every step a squirrel takes requires an update (exchange of information) between the squirrel and the cell its in.

Since we know we'll need to communicate information between a squirrel and the cell it's located in at every iteration it makes sense to use geometric decomposition. This seems like the most logical way to avoid the unnecessary communication between cores. With this pattern we will only need to communicate between cores when a squirrel moves from a cell which belongs to the domain of one processor to the cell in the domain of another. This appears to reduce communication by a considerable amount considering that (depending variably on the size of cells) a relatively small number of squirrel steps will result in it entering a different processors domain.

We have a clear timeline for the problem where we have steps, or iterations. This is clearly a linear problem where we do not use recursive data. Thus our analysis so far suggest the use of domain (or geometric) decomposition. This is very common in simulations in time and space.

It is important to consider load balance. It is likely that during the simulation the majority of the squirrels will be in a minority of the cells and therefore acted on by a minority of the PEs. However, since the squirrels move randomly throughout the landscape it is expected that a good load balance shall be maintained throughout most of the runtime.

# 2 Spread of disease by short range interaction between cells

The biologist want to model the spread of disease into neighbouring cells through means other than squirrels. This is a short range interaction between neighbouring cells. Similarly to the squirrel case, using a domain decomposition shall minimise communication between PEs. Only PEs with adjacent domains shall need to communicate information. Since we have a short range interaction there is never any need for cells that are not geographically adjacent to each other to exchange information. Thus domain decomposition should make communication easy to implement. The communication can be done via halo swaps between PEs with adjacent cells.

# 3 Additional Patterns

## 3.1 Task Parallelism

As discussed above, the tasks involved in this program are short and not computationally expensive. Tasks will be completed quickly on each PE which shall make good load balance very difficult to achieve. Furthermore, it was not possible to create tasks that could be completed entirely concurrently (we need to know populationInFlux and infectionLevel to decide if squirrel reproduces or dies). Therefore we will require PEs to communicate information very regularly and is therefore an inappropriate pattern for this problem.

## 3.2 Pipeline

It is impossible to implement an efficient pipeline for this problem since has short tasks and evolves in time. Our pipeline would have to be only as long as one iteration. A pipeline requires all PEs to communicate with at least one other process and therefore only long pipelines are efficient therefore we can conclude that this is clearly a very inappropriate pattern for this problem.

# 4 Language

In the model that we have described in answer to question 1 we only need to communicate between processes when a squirrel moves from a one cell to another which is operated on by a different process. This should not be very often and should be done by an explicit MPI communication between processes.