**Parallelization**

**Workflow**: The workflow as implemented, for parallel execution of the shallow water wave simulation using domain decomposition is as follows:

1. The original board is divided into square tiles/blocks
2. Each block is padded on all 4 sides with layers of ghost cells as shown in the figure.
3. Each processor is given its own tile padded with ghost cells and works in parallel by time stepping on its own tile.
4. A single processor initially performs the ghost cell updates for the entire board, i.e. updating ghost cells values by copying out rows from left and right and then top and bottom, by wrapping around the board boundaries. Now, the processors are ready to compute their own time step advances in parallel.
5. Each processor also computes the local wave speeds ‘cx’ and ‘cy’ within their own tile and write the values to a shared array, indexing the array by their own unique thread identifiers which range from 0 to number of threads - 1.
6. At the end of ‘n’ time steps, (‘n’ = number of layers of ghost cells – 3, since a minimum of 3 ghost cells are required to compute one entire time step in parallel without requiring data from adjacent tiles), the processors perform a barrier sync.
7. A single processor traverses through the array of local tile wave speeds and computes the maximum speed among them in both x and y directions, and uses this value to perform the time stepping increment.
8. Once the time steps are completed, the true data values are copied back from the ghost cell padded board into the original board.

**Further Experiments**

**Rectangular Tiling**: We also extended the code to be able to work with rectangular tiles and experimented with it, but found that it did not behave as expected. The boundary conditions and ghost cell updates typically change when we have rectangular tiles and it creates complexities and produces strange locally tile wise shifted wave simulation results.

**Blocking at the Driver Level**: We also created an entire implementation of domain decomposition where we divided our original n x n grid into four n/2 x n/2 tiles at the driver level. We then padded layers of ghost cells, and then instantiated these 4 tiles by their own Physics class and allowed these 4 simulations to run in parallel by doing a barrier synchronization between time steps, in order to communicate between processors and to update ghost cell values.

**References**

<https://github.com/amirajdhawan/water> : Held meetings with him to understand their group’s approach, brainstormed and reasoned about potential further optimizations.