Parallelization of a simple Cellular Automaton

The project consists of the parallel implementation in MPI (and OpenMP) of a Cellular Automaton or other structured grid numerical models (es: Predator - Prey, Heat Equation, Sandpile model, Wator, Game of Life, etc.) according to the suggestions given in class. Moreover, it is required to write a report on the used methodology, performance tests, etc. Remember that it is a good idea to implement a first (sequential) version in C.
The parallelization consists to split the matrix in "slices", allocating each slice on a process. This requires the bordering of each subarray, so that the extreme portions of each submatrix have the data which is physically located on adjacent processes. In this regard, recall that a division into horizontal slices allows to send the borders (rows) of submatrices , simply send and receive, while a vertical subdivision requires sending of columns (thus, use of MPI derived datatypes). This applies to a static allocation: pay attention to a possible dynamic allocation (hard though!), where the borders are not contiguous. In addition, you can also use send / receive non-blocking for the sending of the borders, the update of the "inside" portion of the submatrix, and once "finished" the receiving of the borders (with MPI_Wait), updating of the border portion of the sub-matrix (see Models - lesson examples) .
Tip: It's a standard method to allocate (even statically) the whole array for each process. The MASTER process will ONLY send the data portion of each process (start-end of the sub-matrix) . See the example mpi_heat2D.c of the MPI tutorial!

OSS: Although you can implement the project only with blocking sends and receives, solutions will be positively evaluated that involve the use of "advanced" MPI features, such as virtual topologies, derived data types, non-blocking send-receives, etc. In addition, the use of dynamic allocation for the local sub-matrices will be assessed "positively" for the final evaluation.

Finally, the adoption of a suitable display interface (with Allegro, OpenGL, QT, etc)will be further evaluated in a positive way.