Game of Life parallel implementations

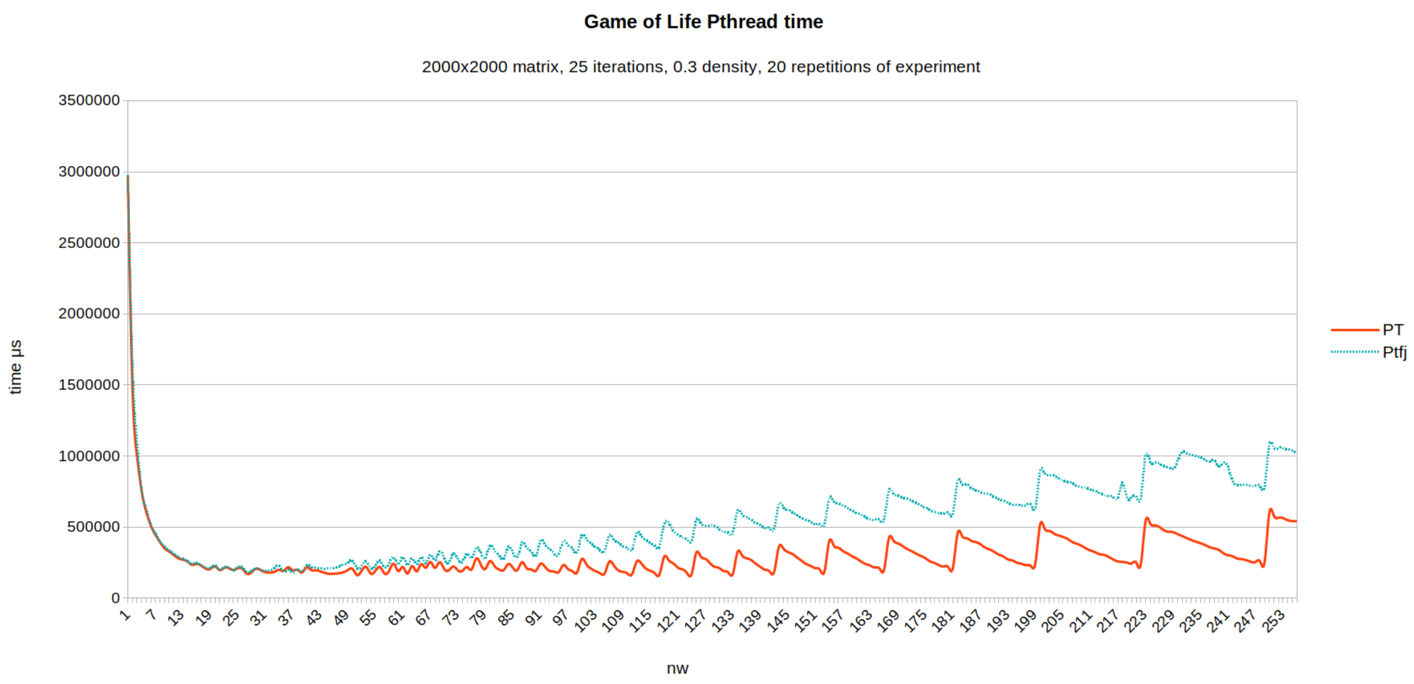
Eleonora Di Gregorio
520655

To implement a parallel version of Game of Life I've realized an abstract class **GameOfLife** and five different concrete extensions that differ one each other for the implementation of the method play. In particular:

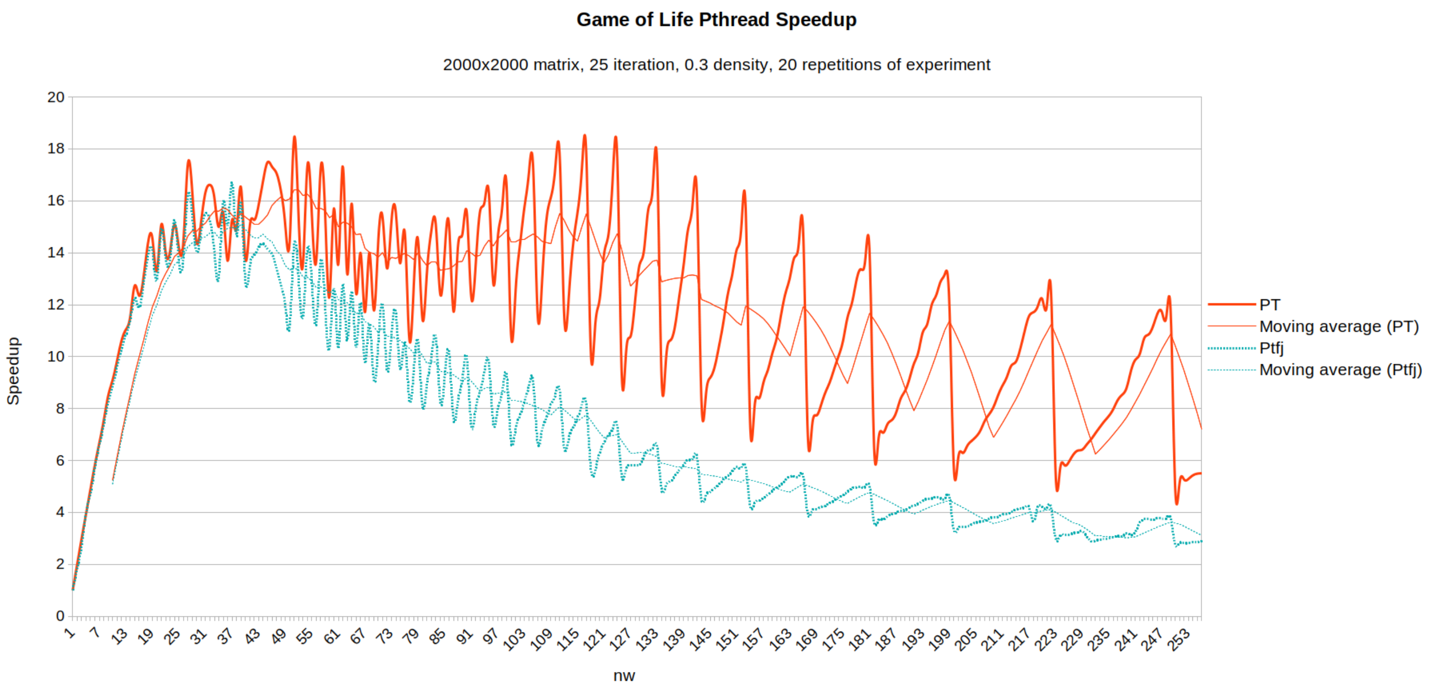
- **GameOfLifeSeq** provides a trivial sequential implementation which scans all the cells of the grid through two nested for. The computed value is stored in a new grid and then grids are swapped.
- **GameOfLifePT**, a pool of thread is created according to the input parameter, and at each iteration, each thread computes the set of rows or part of it (if rows are less then workers), according to its concern. Two condition variables are used to coordinate operations of swap and passage to the next iteration.
- **GameOfLifePTfj**, it's just an experiment in which all threads are created at each iteration.
- **GameOfLifeOMP** provides the sequential implementation extended with "#pragma omp parallel for num_threads(nw) schedule(auto)" , so all is managed by OpenMP.
- **GameOfLifeOMPChunk**, similar to **GameOfLifeOMP** but I have fixed the chunk size and the schedule is static.

Comparisons of Pthread implementation:

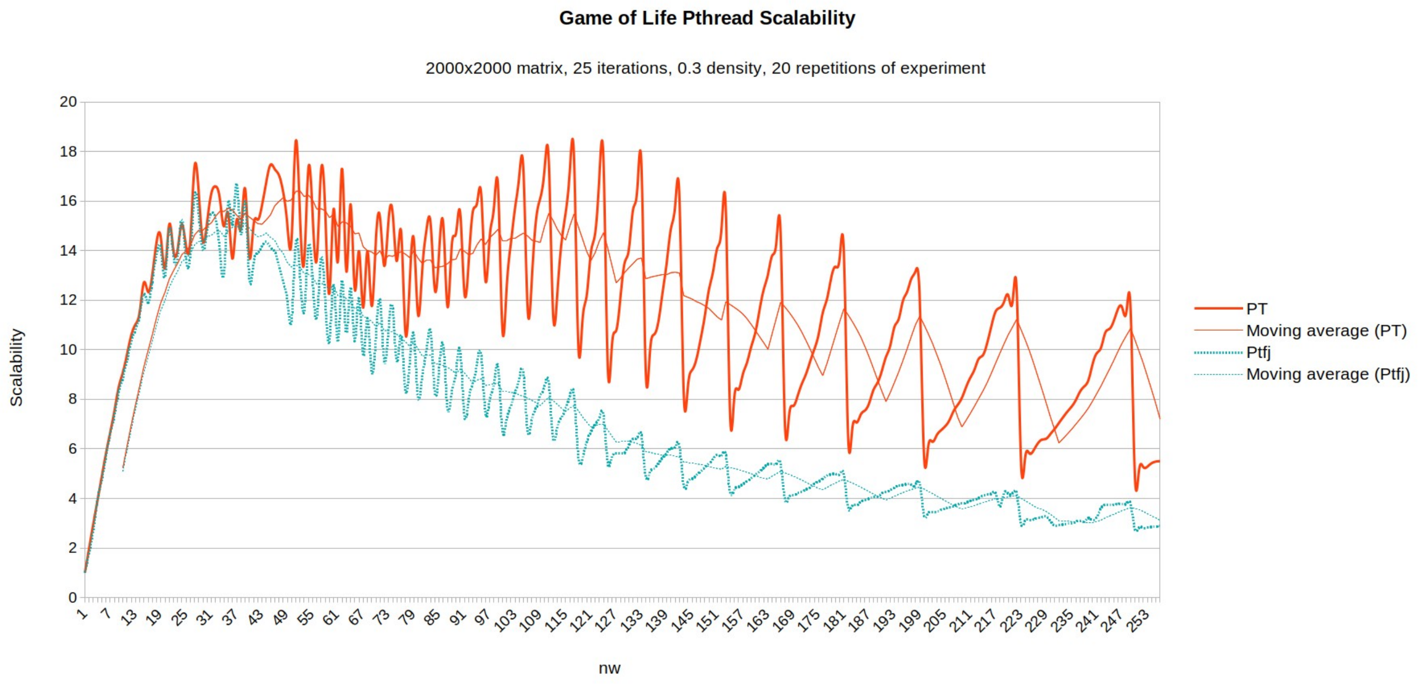
As expected the **GameOfLifePT** has better time performance then **GameOfLifePTfj**.



Speedup is linear up to about 15 workers, it slowly increases up to about 50 workers and then decrease with an interesting pattern which I'm still investigating.

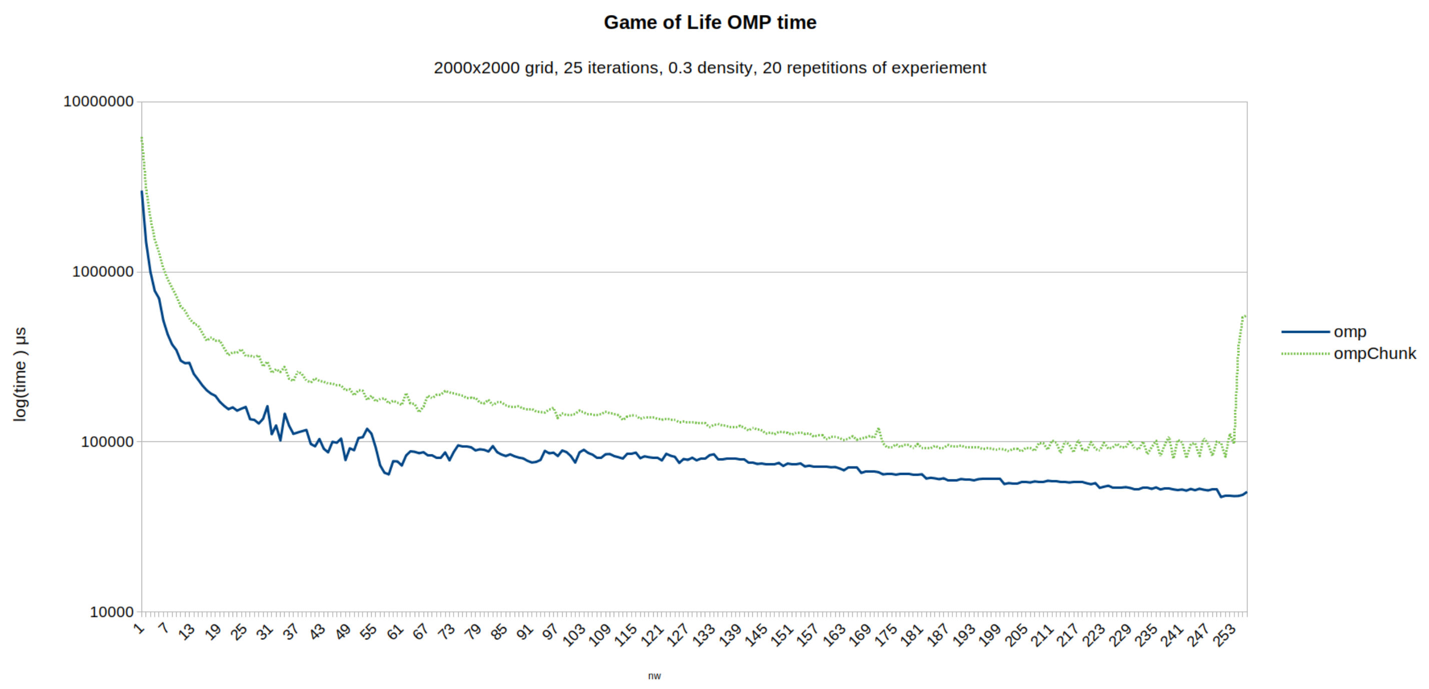


Scalability follows the same trend of speedup, indeed there is a little difference between the sequential time and the time achieved with one thread.

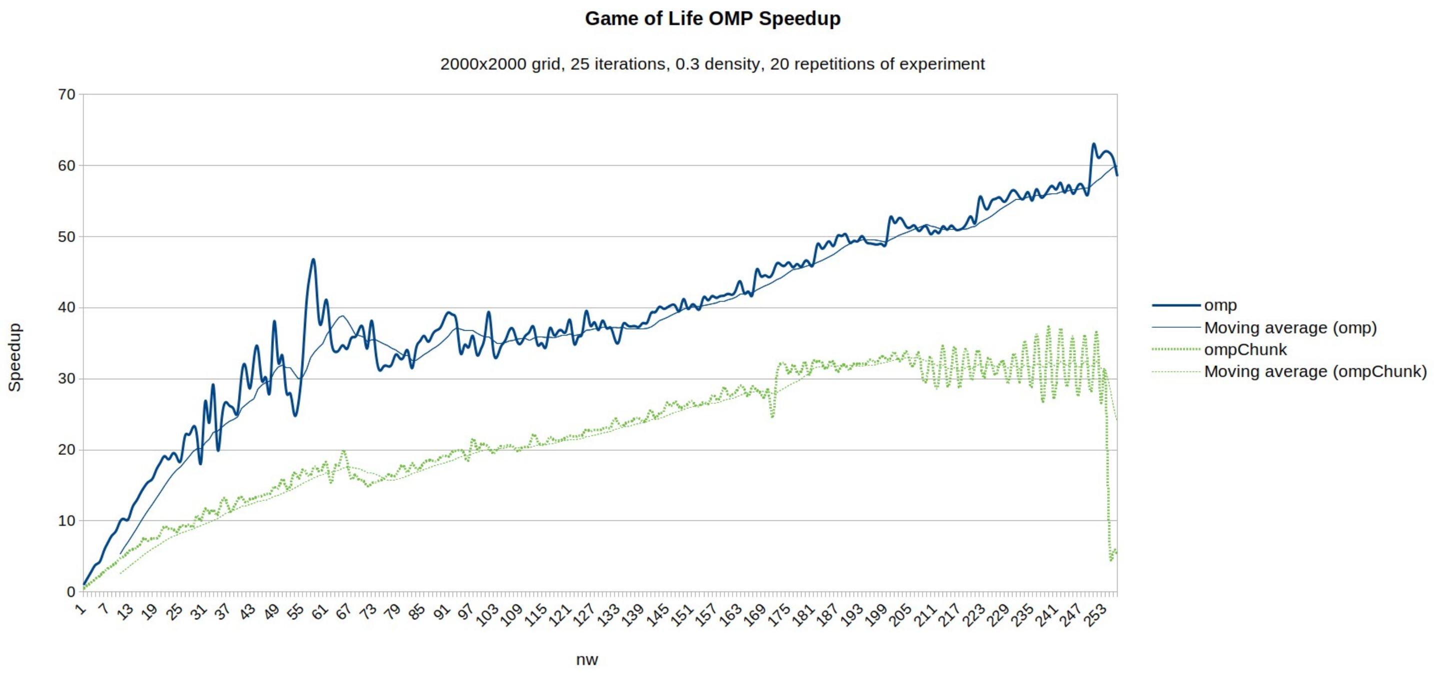


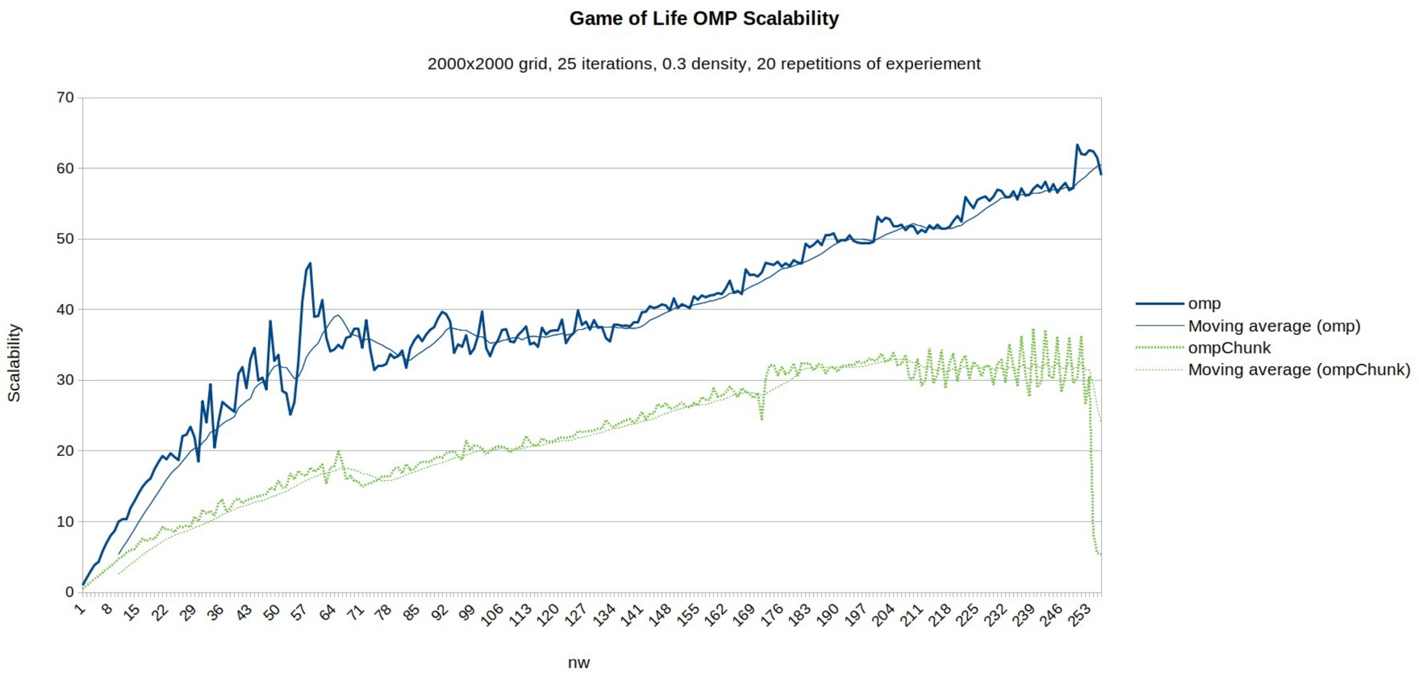
Comparison of OMP implementation:

GameOfLifeOMP has better time performance then **GameOfLifeOMPChunk**, due to the fact that the computation of each task is less heavy.



Speedup and Scalability graphs follow.





Code

Code of classes is available attached to this file, to run an example with **GameOfLifeSeq**, **GameOfLifePT** and **GameOfLifeOMP** just compile gof.cpp file.
Parameters needed are:

- Number of iteration (mandatory)
- Random number generator seed (mandatory)
- Number of rows (mandatory)
- Number of columns (mandatory)
- Parallelism degree (mandatory)
- Densitity of grid (default value 0.3)
- Print the grid after each iteration (default value false)
- Print the grid at the end of all the iterations (default value false)

All the experiments have been runned on Xeon PHI server after compiling with g++ 9.2.0 version and the following flag : -fopenmp -O3 - DNDEBUG -ptrhead.

The code, including that for the previous benchmark, and this report are available on github here: https://github.com/eleonoradgr/GameOfLife