Parallelizing C++ Game of Life with OpenMP

Parallel Computing Laboratory

Giacomo Orsucci

February, 2026

UNIVERSITÀ DEGLI STUDI FIRENZE

Da un secolo, oltre.

Project objective

- Implement a first sequential version of cellular automata simulation Game of Life.
- Experiment various parallel versions using OpenMP.
- Evaluate the impact given by parallelization, cache tiling, SIMD, etc etc.

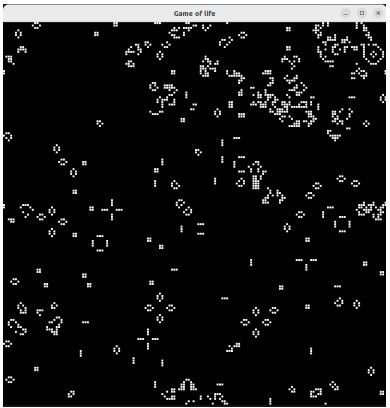


Figure: Graphical Game of Life simulation.

Game of life rules

Four main rules based on counting neighbors of a 3x3 grid centered on cell:

- **Underpopulation**: A live cell with fewer than two live neighbors dies.
- **Stays Alive**: A live cell with two or three live neighbors stays alive.
- **Overpopulation**: A live cell with more than three live neighbors dies.
- **Reproduction**: A dead cell with exactly three live neighbors becomes alive.

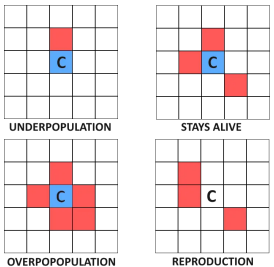


Figure: Four main rules of Game of Life.

Different implementations and experiments

Sequential

```
1
2     for each row:
3         ...
4         for each column:
5
6             count neighbors around cell (i,j);
7             apply Game of Life rules to cell (i,j);
8             ...
9
```

UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

Different implementations and experiments

Parallel

```
#pragma omp parallel for collapse(2)
for each row far from borders:
    for each column far from borders:

        count neighbors_noBorder around cell (i,j);
        apply Game of Life rules to cell (i,j);
        uses branchless logic;

for each row and column near border:

        count neighbors_Border around cell (i,j);
        apply Game of Life rules to cell (i,j);
        uses branchless logic;
```

Different implementations and experiments

Tiled

```
#pragma omp parallel for collapse(2)
for each ii in TILE_SIZE:
    for jj in TILE_SIZE:

    count neighbors_Border around cell (i,j) and
    apply Game of Life rules to cell (i,j) differently if
    is near or far from borders.

    Uses simd and branchless logic.
```

Different implementations and experiments

SIMD optimization

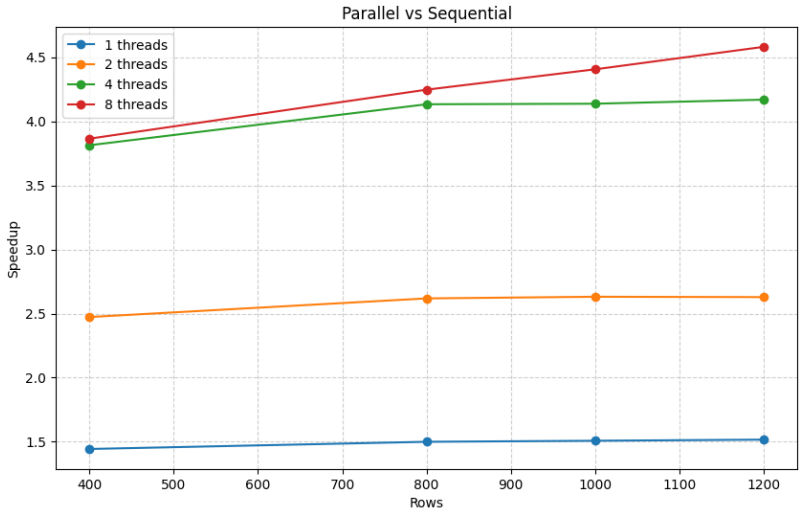
```
#pragma omp parallel for
for each row far from borders:

    count neighbors (using precalculated indexes)
    and apply Game of Life branchless logic really
    optimized for classic Game of Life SCAN_SIZE=3.

    for each row near top and bottom borders:

        count neighbors_border and apply
        Game of Life branchless logic.

    for each column near left and right borders:

        count neighbors_border and apply
        Game of Life branchless logic.
```

Machine Specifications

- **CPU**: AMD Ryzen 7 3700U, 4 cores, 8 threads, base clock 2.3 GHz, max boost clock 4.0 GHz.
- **Cache**:
  - L1 Cache: 96 KB per core.
  - L2 Cache: 512 KB per core.
  - L3 Cache: 4 MB shared.
- **RAM**: 20 GB DDR4 (1 x 4 GB SODIMM DDR4 + 1 x 16 GB Row of Chips DDR4) at 2400 MHz.
- **OS**: Ubuntu 22.04 LTS.
- **Compiler**: GCC 11.4.0.
- **OpenMP**: version 5.2.

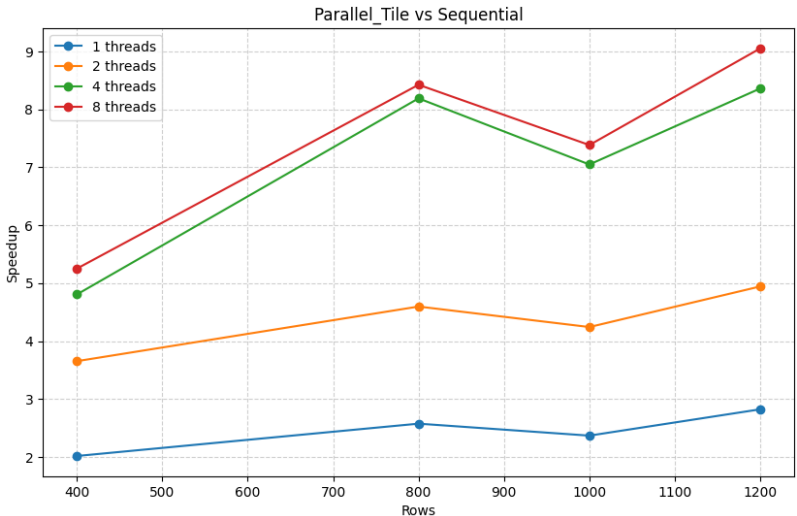
Experimental results

Parallel vs Sequential



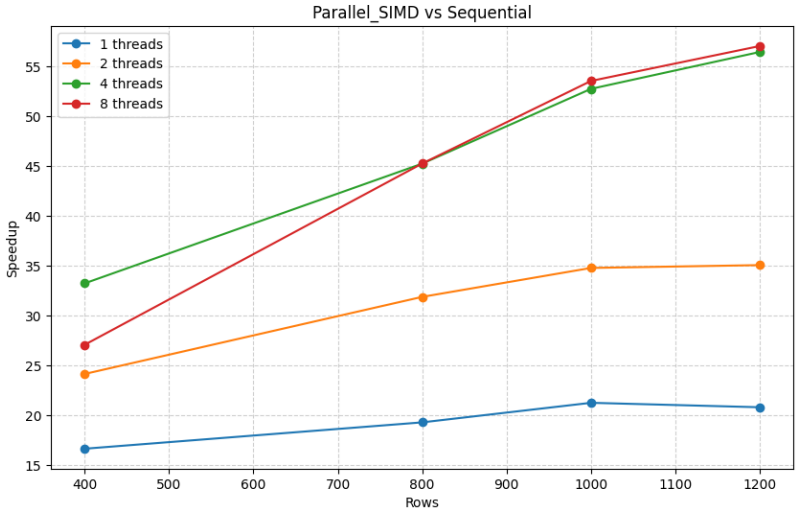
Experimental results

Tiled vs Sequential



Experimental results

SIMD optimized vs Sequential



Conclusion

- OpenMP parallelization is effective, but its full potential is locked without proper code optimization.
- The experimented cache tiling is a good optimization.
- Hyper-Threading proved to be not so beneficial, significantly increasing contention in the Memory-Bound scenario.
- The maximum SIMD optimization has achieved impressive results.