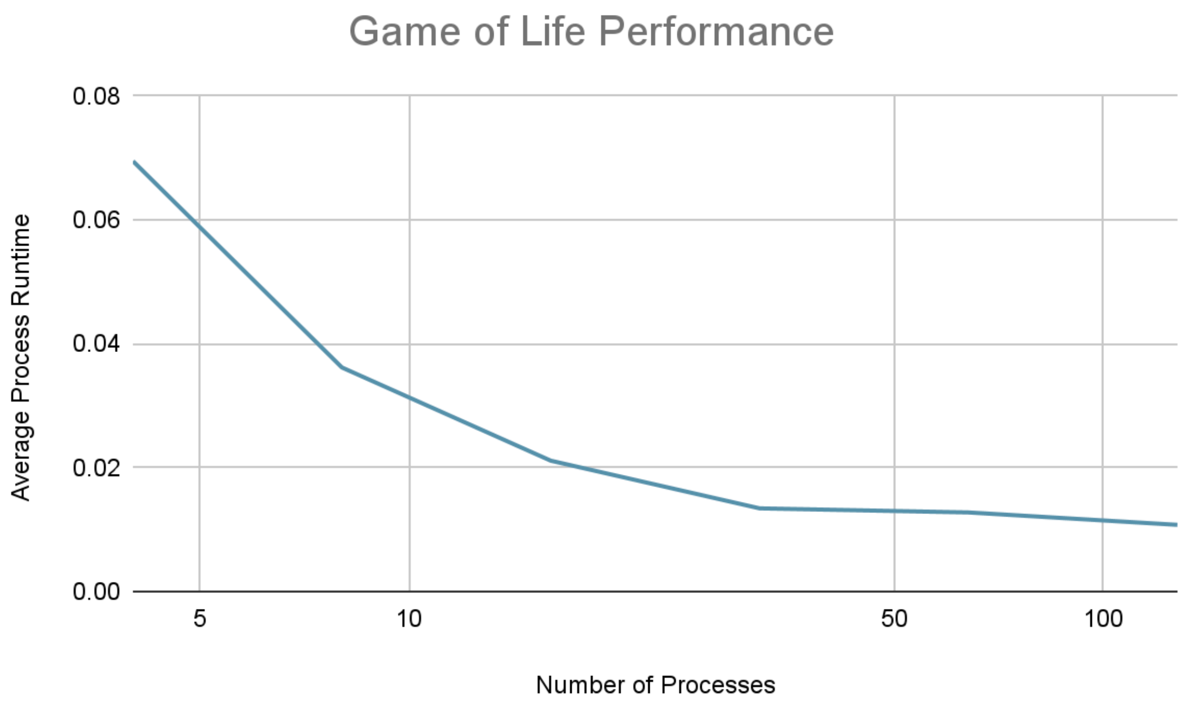

Initial Data Distribution: Rank 0 reads the full input data into a 2D array (`full_life`), then it uses non-blocking `MPI_Isend` calls to send the appropriate rows to the other processes. Each process, including rank 0, stores its portion of the data in its local grid (`life`), with an additional row at the top and bottom (ghost rows) for exchanging boundary data with neighboring processes. The processes then use non-blocking `MPI_Irecv` to receive their grid portion. Rank 0 waits for all non-blocking sends to complete using `MPI_Waitall`, ensuring that the data is fully distributed before the simulation begins.

Analysis: The performance results demonstrate a clear and expected trend: as the number of processes increases, there is considerable speedup compared to the serial code. For instance, with 4 processes, the parallel execution time is reduced by about 95% compared to the serial execution. We achieve a maximum speedup of over 130x with 128 processes. The decrease in runtime is not perfectly linear due to communication overheads between processes, especially beyond a certain point (e.g., 64 and 128 processes). This outcome is consistent with our expectations, given that we used a 1D decomposition strategy and non-blocking communication to mitigate the costs of message passing and load imbalance. The diminishing returns suggest that at some point, communication overhead starts to outweigh the benefit of further parallelization.