David Pérez // NIA: 241375 Eduard Rodríguez // NIA: 242710

Lab 2: MPI

Exercise 2:

2.4: Study the strong speedup:

➔ 1 process vs 1, 2, 4, 8 and 16 threads:

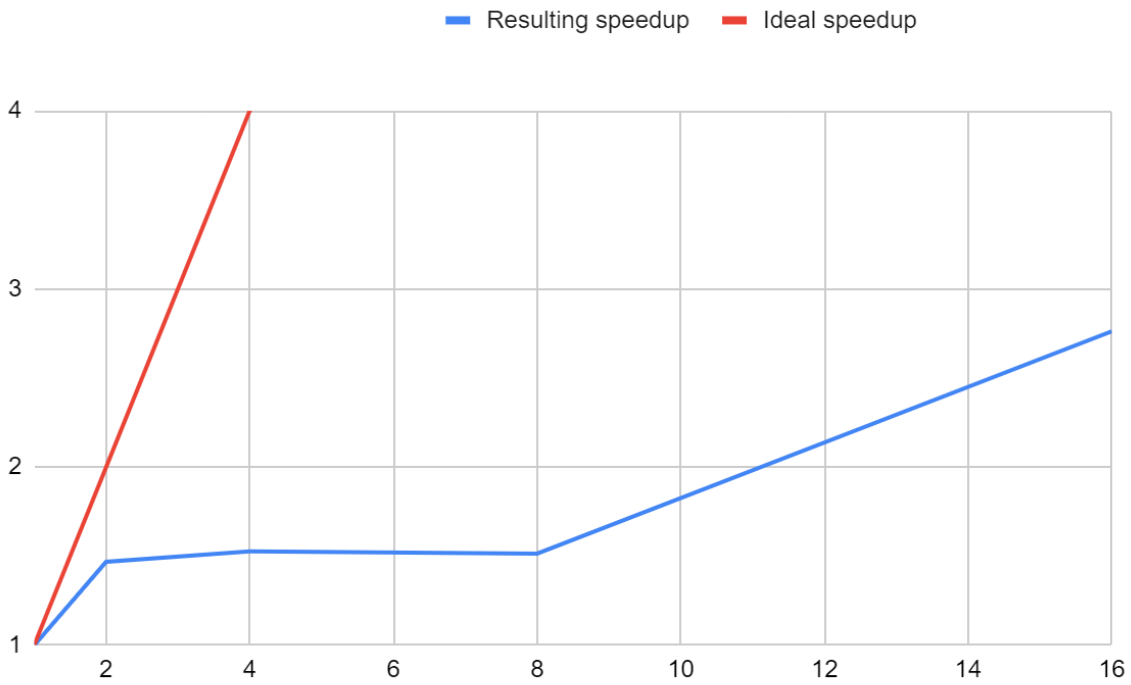


figure 1
x axis: nº of threads / y axis: strong speedup

As seen in figure 1, the scalability of the number of threads is good.

➔ 1 thread vs 1, 2, 4, 8 and 16 processes:

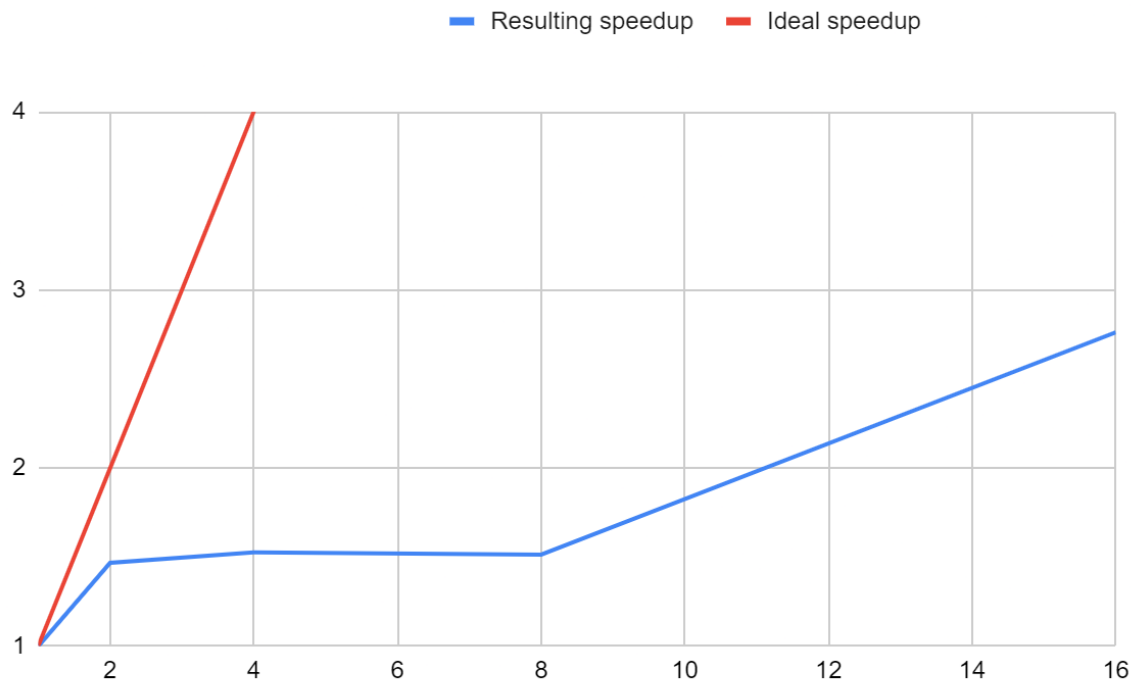


figure 2
x axis: nº of processes / y axis: strong speedup

When increasing the number of processes, we get an extremely similar result.

This equivalence is due to the fact that both implementations parallelize the code in the same manner. The only difference being that processes create much more overhead that threads, which isn't taken into account in the elapsed time computation hence the equivalence. We can even see (when that the

2.5: Study the best combination of processes and threads:

After trying both combinations of processes and threads, we have found that the case in which we set 2 processes and 12 threads gives us a time of around 0,017 while the other combination, 4 processes and 6 threads gives us a time of around the same values as for the former case. This, once again, is due to the similar nature of threads and processes (without considering overhead).

Exercise 3:

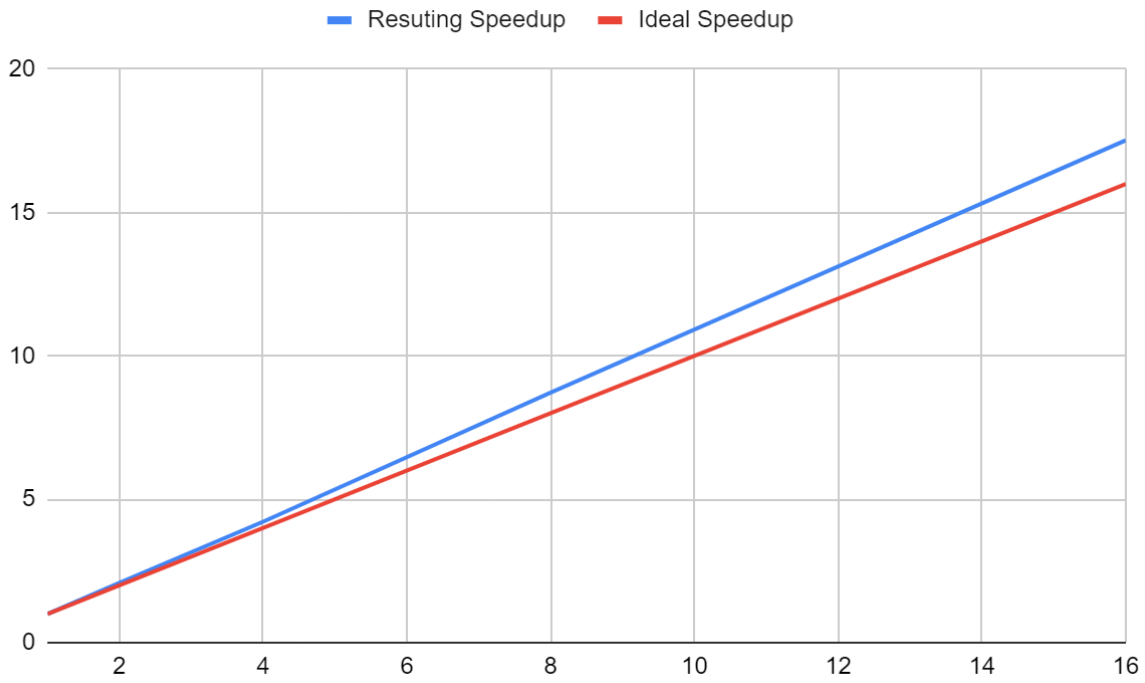


figure 3
x axis: nº of processes / y axis: strong speedup

In this case, we have an incredibly good speedup for all processes.

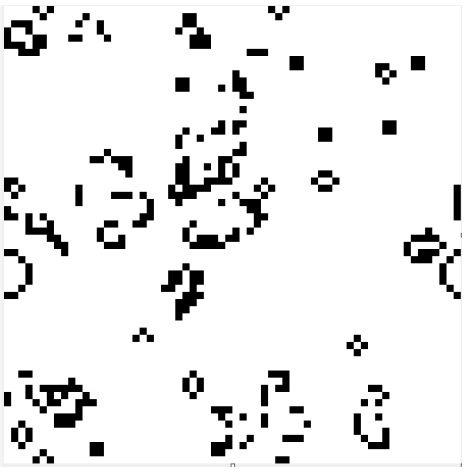
Exercise 5:

6 Provide the initial and final bitmaps for life1.bin and life2.bin with 1 and 8 processes.

Life1.bin Initial and final bitmaps:

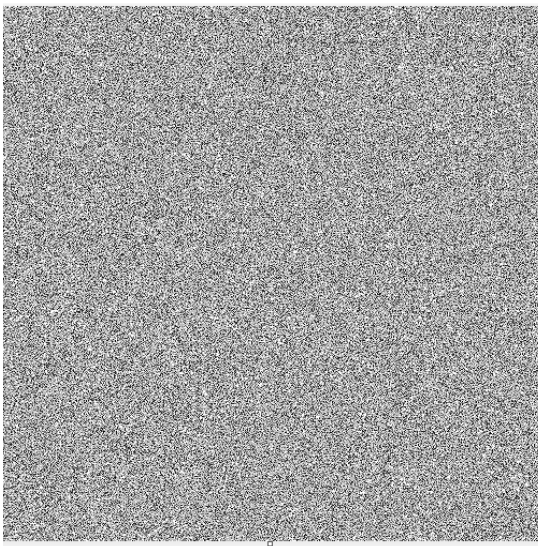
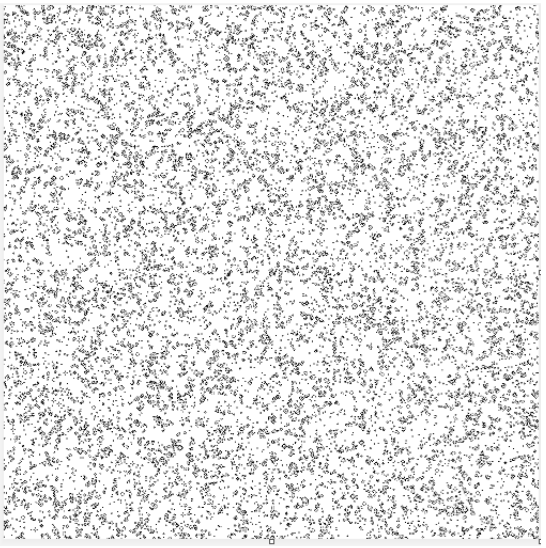
- ☐ 1 process:

☐ 8 processes:



Life2.bin Initial and final bitmaps:

☐ 1 process:

☐ 8 processes: