Charles Dufour
257587
charles.dufour@epfl.ch

Parallelized 2D Poisson solver with MPI

P&HPC, MATH-454

2022-11-29

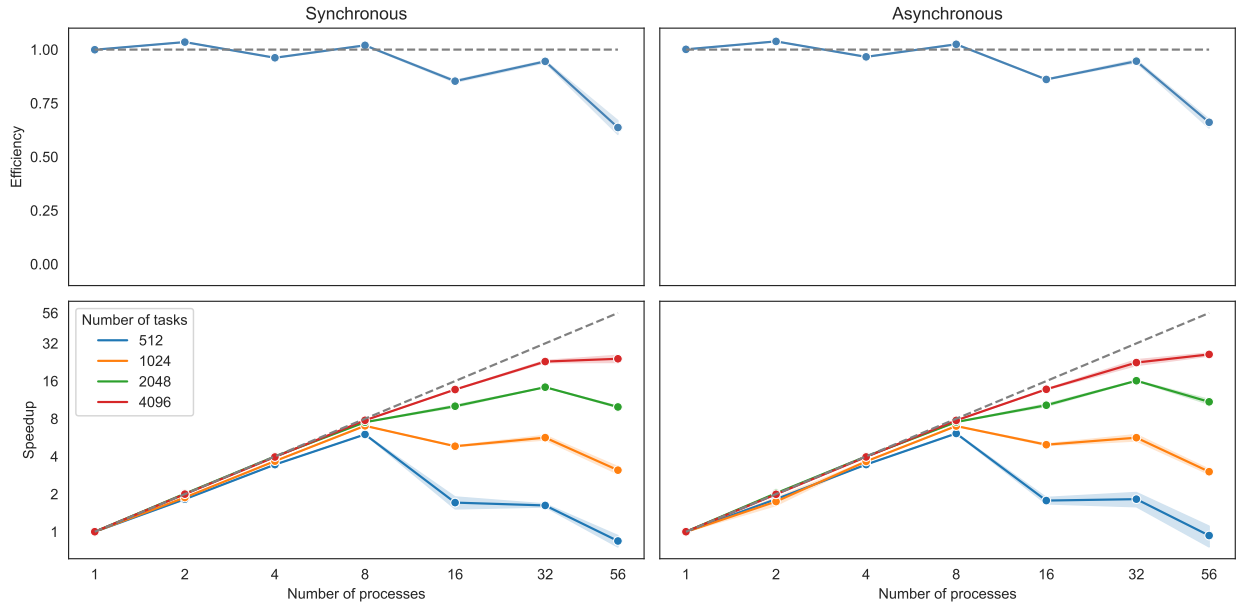

Figure 1: Strong and weak scaling results from the parallelized version of the 2D Poisson solver with MPI.The top row shows the weak scaling, and the bottom shows the strong scaling. The columns indicate the different communication methods used. The grey dotted lines represent the best efficiency (Gustafson's law) and speedup (Amdahl's law) theoretically attainable

- *Weak scaling* (row 1 of Figure 1): we do not have perfect weak scaling: as we increase the number of processes beyond 8 the efficiency is reduced. The drop-off could be explained by the time spent communicating between processes at each iteration (synchronization overhead)[1].
- *Strong scaling* (row 2 of Figure 1): if the size of our task is too small for the number of processes p we use, the speedup suffers. For example, for a task of size 512, the apex speedup is attained at 8 processes, while 16 processes provide a speedup equivalent to 2.

  Amdahl's law would explain this with the unparallelizable portion of the code. However, we do not have a non-parallelizable part in the initialization or computation of the solver, nor do we have I/O operations. The blocking communications between processes could explain the reduction: when the number of processes is sufficiently significant compared to the problem size, synchronisation is the limiting factor. The load imbalance can also be a factor in explaining the irregularity of the curves.

Note on the framework Speedup is defined as the $s_p = t_s/t_p$, where t_s is the computational time for serial and t_p for parallel using p processors with constant total workload. Efficiency is defined by the ratio $e_p = t_1/t_p = s_p/p$, where the workload per processor is constant.
The first row in Figure 1 was computed by letting the number of tasks grow with the number of processes used for the computation: for p processes, the problem size was $4096 \times p^{1/2}$. The solver ran for 50 iterations for all experiments. Figure 1 shows the results of two repetitions for each experiment.

[1]The fact that the synchronous and asynchronous versions have the same behaviour can either indicates that the communication time is the limiting factor or that the asynchronous communications still suffer from waiting times due to not optimal implementation (probably the last hypothesis).