HPC, Parallel Programming: Homework 2

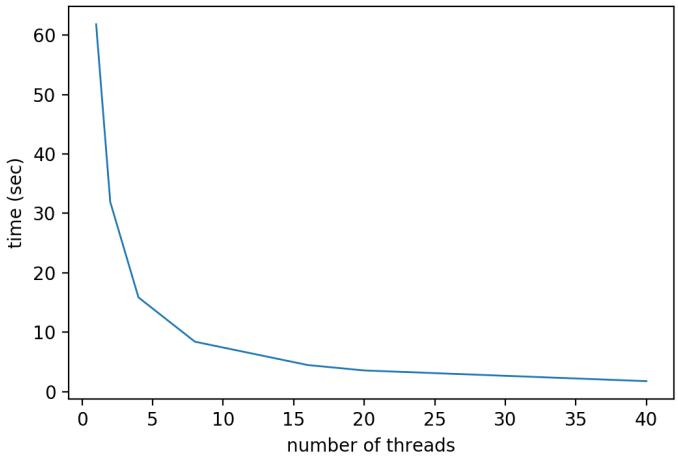
Simone Amadio

General informations

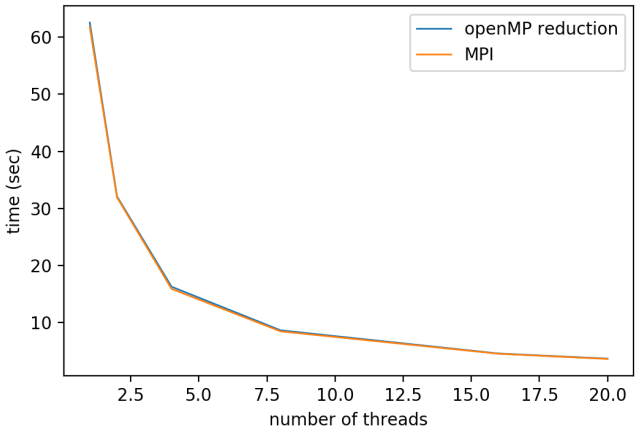
For clarity and brevity, the results will be presented mainly as graphs. The code and files containing the results are provided separately in this same folder. Only peculiar results will be analyzed singularly.

1 OpenMPI parallelization

In this exercise I tested the different behaviour of MPI with respect to openMP approach. In order to do so I rewrote a code for evaluating π using a series and submitted bash scripts to Ulysses. A time measurement has been taken for each number of threads.
I run bash scripts in order to test the scaling of an MPI application using 1,2,4,8,16,20 and 40 threads. For each instance of the problem I did 2 tests to ensure result stability. Noting that the single result for each number of threads presents little to no variation ($\leq 0.1\%$ at most), I used a single value between the ones generated.
Using N=10^{10} points for evaluating π, the following scaling curve is obtained



MPI Scaling curve for N=10^{10}



Confronting openMP with MPI

This picture is to be confronted with the curve obtained through the openMP multithreaded approach used on homework 1.
As we can see, the two results are overall equivalent. This is probably due to the nature of the program: based on Ahmdal Law, we know that the scaling of a parallel application is bounded by its serial part; in this case, the serial part is almost negligible with respect to the parallel component.
As done in homework 1, an analysis of the overall speedup has been conducted. The MPI speedup seems to be negligibly greater than the openMP reduction method (relative difference is $\leq 0.03\%$).
This results underline that the message passing has been effectively reduced, causing the communication overhead to be negligible. The last measures of time, obtained by using 40 threads, also support the statement that an opportune program can have nearly linear speedup even when running on multiple nodes, if limitations caused by the network are not encountered.

Code and file references

- Results are obtained through the script `script.sh`. The code in `pi.c` has been used to obtain an executable.
- Results and times are contained in the `results.txt` file. It contains the estimate of π and the time taken twice for each number of threads. The results are stored in ascending order with respect to the number of threads.