Università della Svizzera italiana

Institute of Computing CI

High-Performance Computing **2022**

Student: SIMONE TARENZI Discussed with: VEDANG NAIK, ISIN SU ECEVIT

Solution for Project 4

Due date: 23.11.2022, 23:59

1. Ring maximum using MPI [10 Points]

First I used a simple way to get what rank the left and right neighbours of each node are.
The right node is given by the expression (my_rank+1)%size: in our case for example, node 3 will have as its right neighbour 0, because (3+1)%4 is 0.
The left node instead is given by the expression (my_rank-1+size)%size, meaning that node 0 will have as its left neighbour node 3, since (0-1+4)%4 = 3.
Implementing the actual comunication between nodes was pretty straightforward: each node, after making the operation to calculate its own max, goes into the for loop in which it sends that max to its right neighbour, and receive the max of its left neighbour. If the max it receives is higher than its own, its max becomes the new max, and so it will send that to its right neighbour. At the end of the loop, each node will have the max between every node, which in our case is 6.

2. Ghost cells exchange between neighboring processes [15 Points]

This exercise was a bit trickier to understand.
For starters, the documentation found online for the MPI_Cart_shift command was actually wrong, since it reversed how to find the left and right neighbours with the top and bottom ones.
After that, I had to create an additional MPI_Datatype. I used the given data_ghost for columns and created data_row for rows.
Rows are easier to create: since the matrices are created left to right, top to bottom, the array of data will get the sequence of doubles as-is.
To create the columns instead, it needs to jump 8 indices to get the next correct number (Fig.1).

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Figure 1: the ghost matrix indices, red dots represent the ones to send

After the datatypes are created, I can send them.
First I send all the data to the neighbours, and then it can be received. Trying to send only one vector at a time and receiving it immediately after creates unwanted results.
The code works with any of the processes, even boundary ones.

3. Parallelizing the Mandelbrot set using MPI [20 Points]

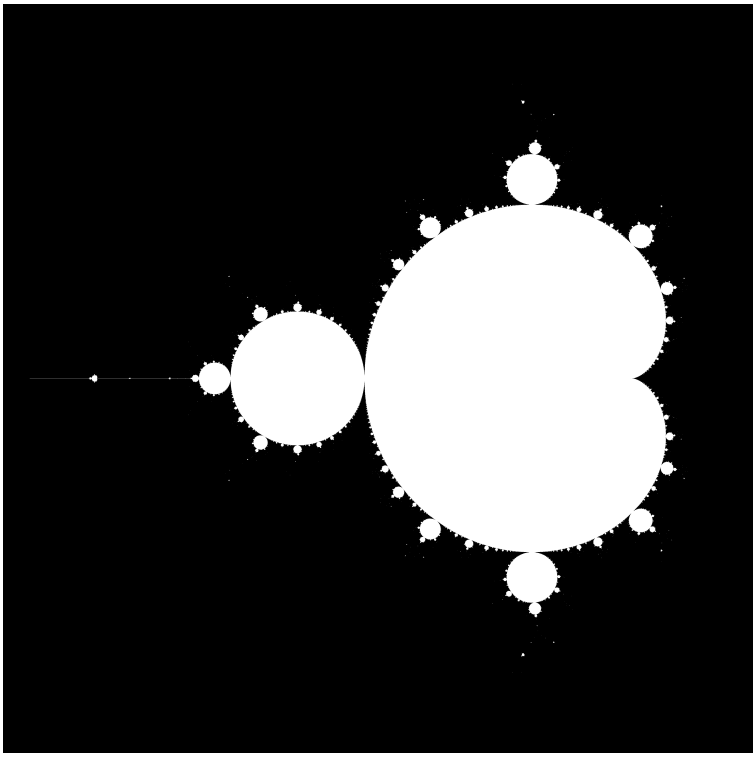

Figure 2: the resulting mandelbrot image

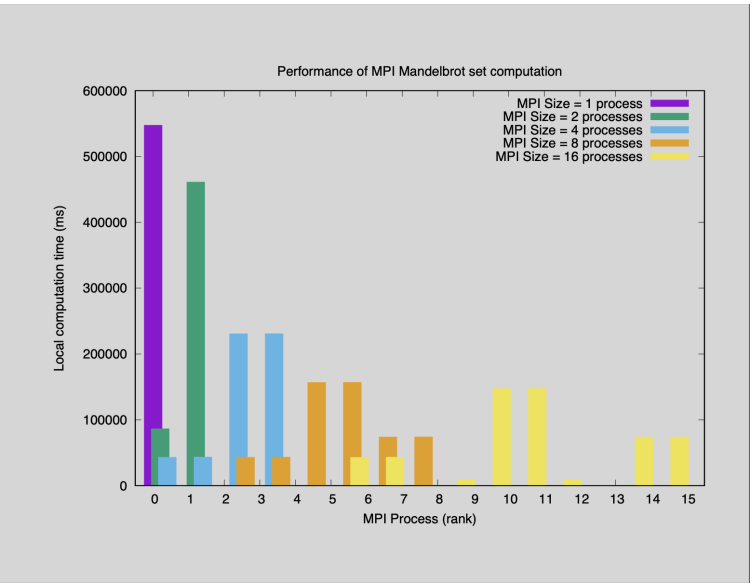

Figure 3: the perf.ps performance graph of the computation divided between various number of processes

The performance is already improved when using 2 processes instead of one, but a really interesting finding is that the computation time isn't evenly distributed between the processes.

4. Option A: Parallel matrix-vector multiplication and the power method [40 Points]

I couldn't implement the parallelization, but the lambda result should at least be correct.

5. Task: Quality of the Report [15 Points]