Università della Svizzera italiana | **Institute of Computing CI**

High-Performance Computing **2022**

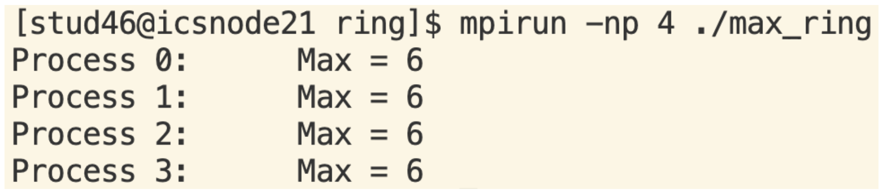
Student: Qianbo Zang Discussed with: Mazio Lunghi

Solution for Project 4

Due date: 23.11.2022, 23:59

1. Ring maximum using MPI [10 Points]

Each process sends a value to its neighbors. If the value received from its neighbors is higher than the current maximum , the maximum will be updated. Otherwise, it remain.

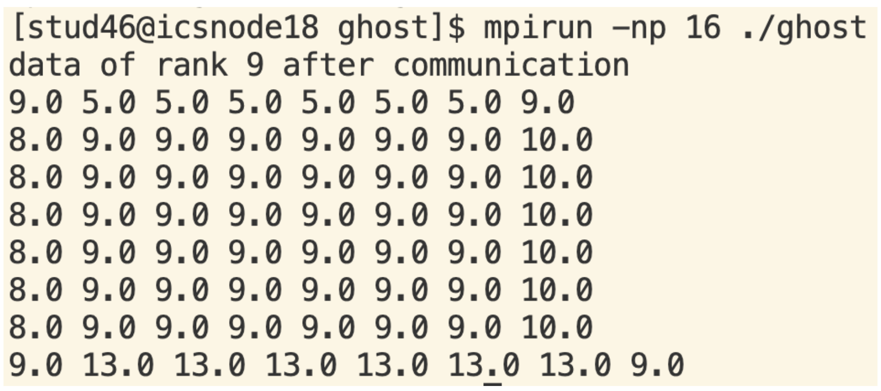


```
[stud46@icsnode21 ring]$ mpirun -np 4 ./max_ring
Process 0:      Max = 6
Process 1:      Max = 6
Process 2:      Max = 6
Process 3:      Max = 6
```

Figure 1: Ring Program

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

Using the Cartesian two-dimensional communicator, I look for adjacent ranks in a circular manner in all dimensions.



```
[stud46@icsnode18 ghost]$ mpirun -np 16 ./ghost
data of rank 9 after communication
9.0 5.0 5.0 5.0 5.0 5.0 5.0 9.0
8.0 9.0 9.0 9.0 9.0 9.0 9.0 10.0
8.0 9.0 9.0 9.0 9.0 9.0 9.0 10.0
8.0 9.0 9.0 9.0 9.0 9.0 9.0 10.0
8.0 9.0 9.0 9.0 9.0 9.0 9.0 10.0
8.0 9.0 9.0 9.0 9.0 9.0 9.0 10.0
8.0 9.0 9.0 9.0 9.0 9.0 9.0 10.0
9.0 13.0 13.0 13.0 13.0 13.0 13.0 9.0
```

Figure 2: Ghost Cells

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

The Mandelbrot Figure is shown below:

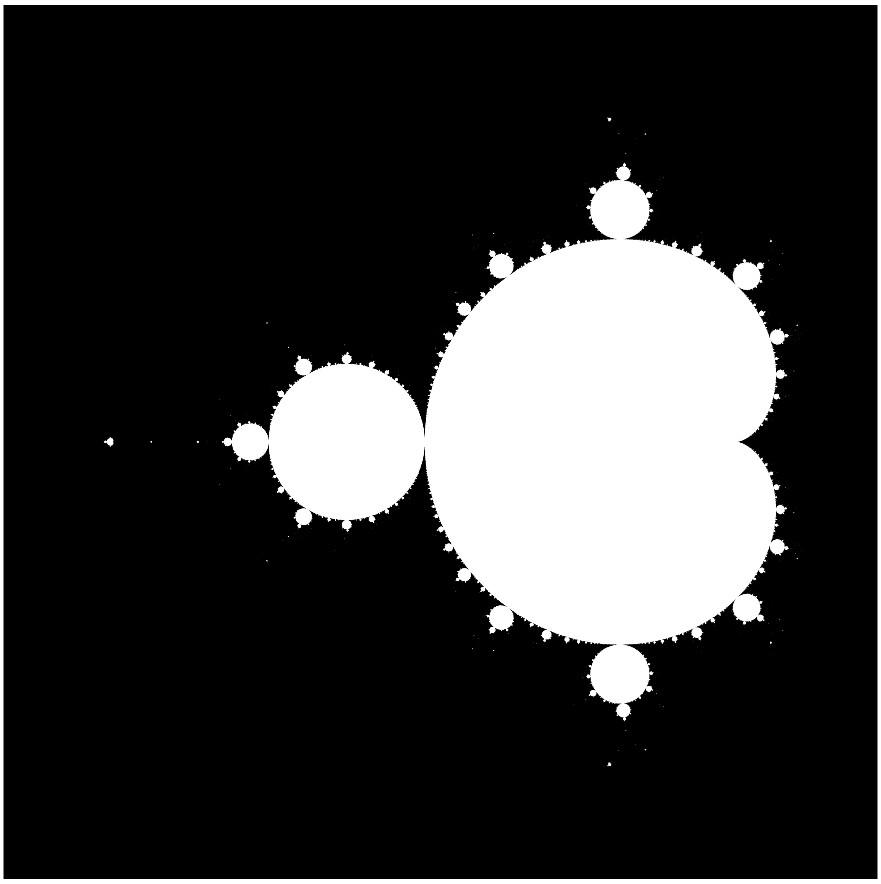


Figure 3: Mandelbrot

When using from 1 to 8 processes, I find that the average of local computational time decrease. But I find that the local computational time by 8 MPI processes is equal to it of 16 MPI processes. I can draw a conclusion that each task have an optimal calculation time, which is corresponding to a specific number of processes. On top of thatif we want to add more processes, the local computational time will not decrease anymore.

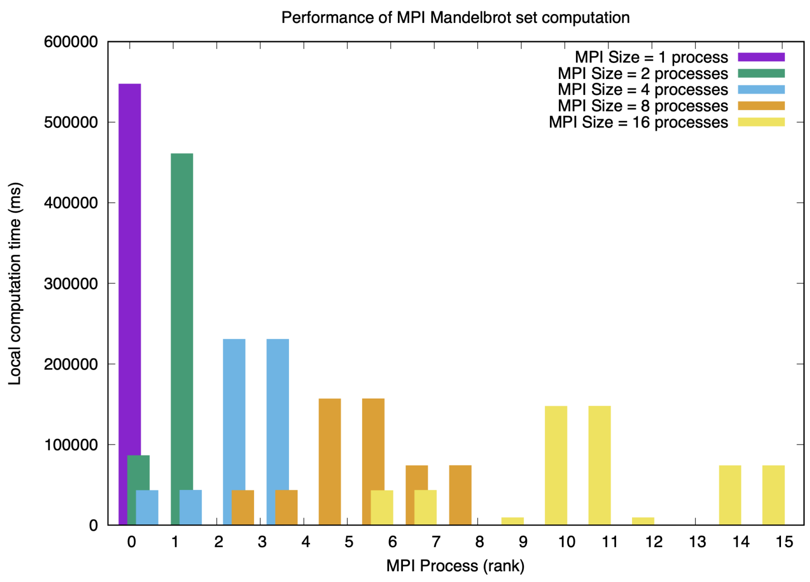


Figure 4: Local Computational Time of Mandelbrot

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

4.1. Strong scaling analysis

According to matrix multiplication theory, the matrices can be divided into small blocks. Then, matrices are decomposed to calculate the product and merged to obtain the final result. So matrix multiplication is suitable to parallelisation.

Acording my picture, before 20 processes, it shows that the walltime decrease when increase the number of processes.

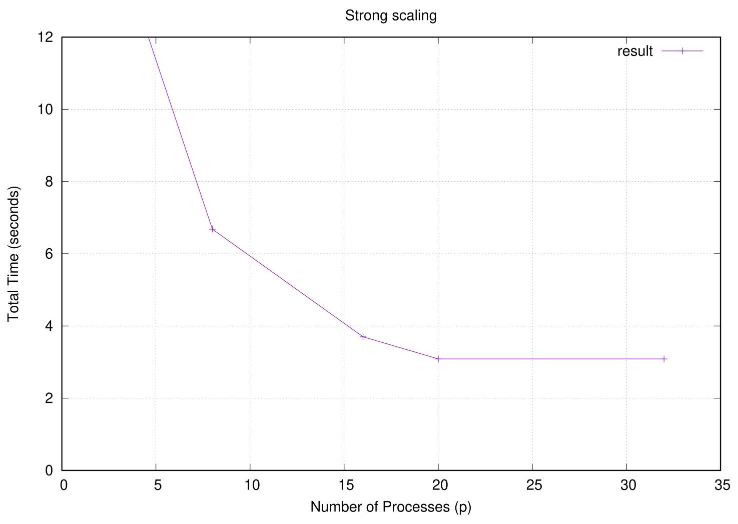


Figure 5: Strong scaling

The parallel efficiencies for a fixed size problem are more than 90 percent until the number of processes goes past 12. After 20 process, the efficiency drops significantly because on our supercomputer, there are only 20 cores available for each node. When more than 20 processes, there will be telecommunication time among each node.

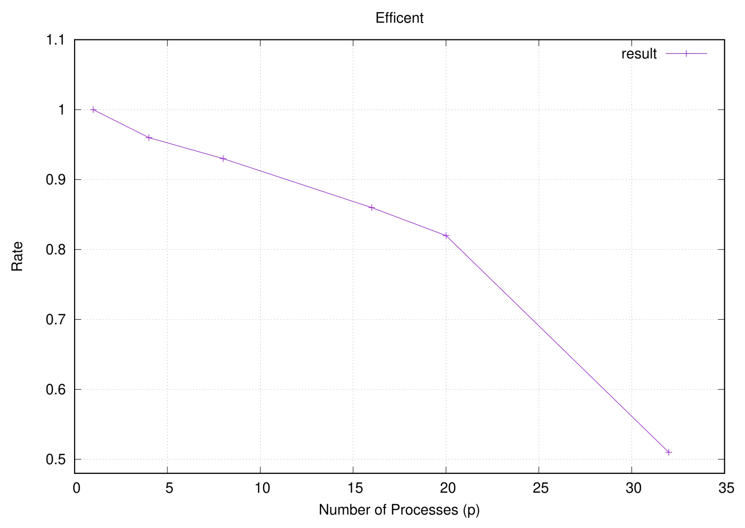


Figure 6: Efficiency

4.2. Weak scaling analysis

I start with p = 1, n = 6400. Then, p = 4, n = 12800...... The execution time only creep up a little as the number of processes increases. The weak scaling result is good compared with the last assignment.

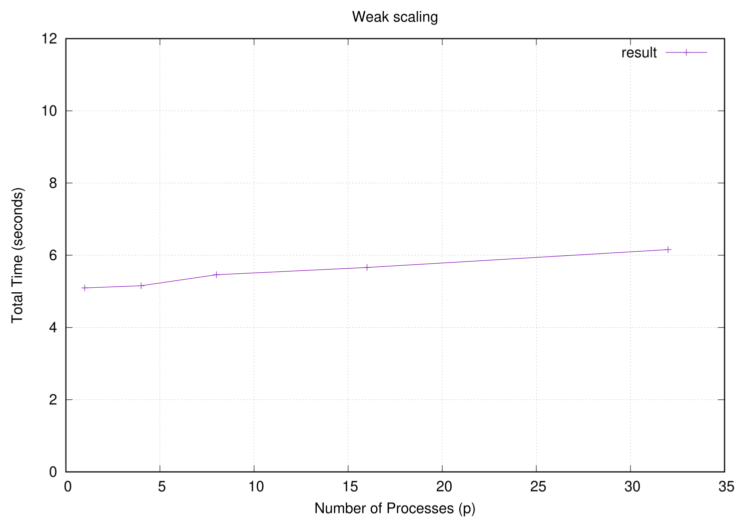


Figure 7: Weak scaling

5. Option B: Parallel PageRank Algorithm and the Power method [40 Points]

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

Each project will have 100 points (out of which 15 points will be given to the general quality of the written report).

Additional notes and submission details

Submit the source code files (together with your used `Makefile`) in an archive file (tar, zip, etc.), and summarize your results and the observations for all exercises by writing an extended Latex report. Use the Latex template from the webpage and upload the Latex summary as a PDF to iCorsi.

- Your submission should be a gzipped tar archive, formatted like project_number_lastname_firstname.zip or project_number_lastname_firstname.tgz. It should contain:
  - all the source codes of your MPI solutions;
  - your write-up with your name project_number_lastname_firstname.pdf,
- Submit your .tgz through Icorsi.