Università della Svizzera italiana

Institute of Computing CI

High-Performance Computing **2022**

Student: Anthony Bugatto Discussed with: Riccardo Giacometti

Solution for Project 4

Due date: 23.11.2022, 23:59

1. Ring maximum using MPI [10 Points]

Using the given specifications the desired output was achieved.

```
[stud07@icsnode37 ring]$ mpirun -np 4 ./max_ring
Process 0:      Max = 6
Process 1:      Max = 6
Process 2:      Max = 6
Process 3:      Max = 6
[stud07@icsnode37 ring]$
```

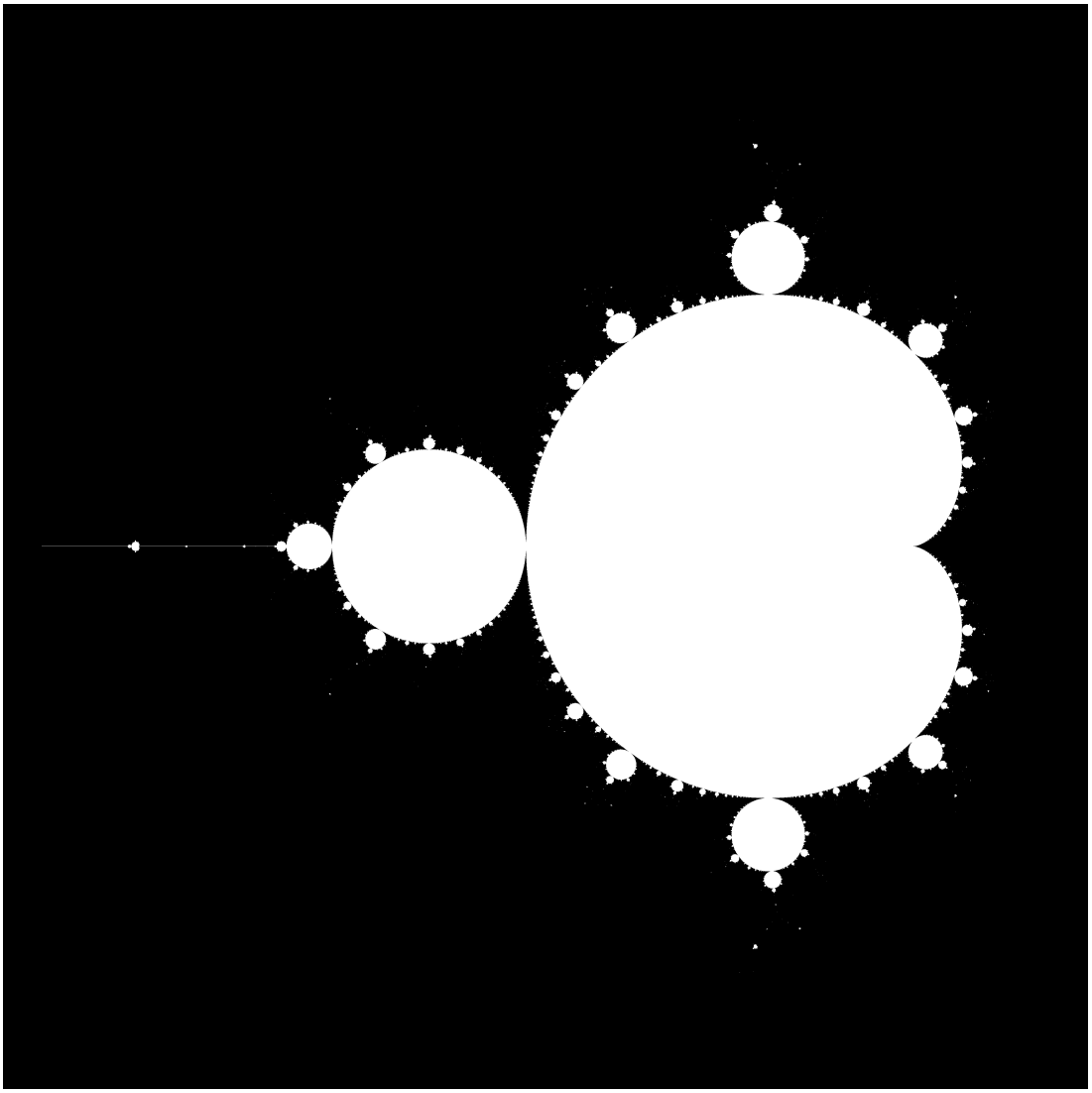
2. Ghost cells exchange between neighboring processes [15 Points]

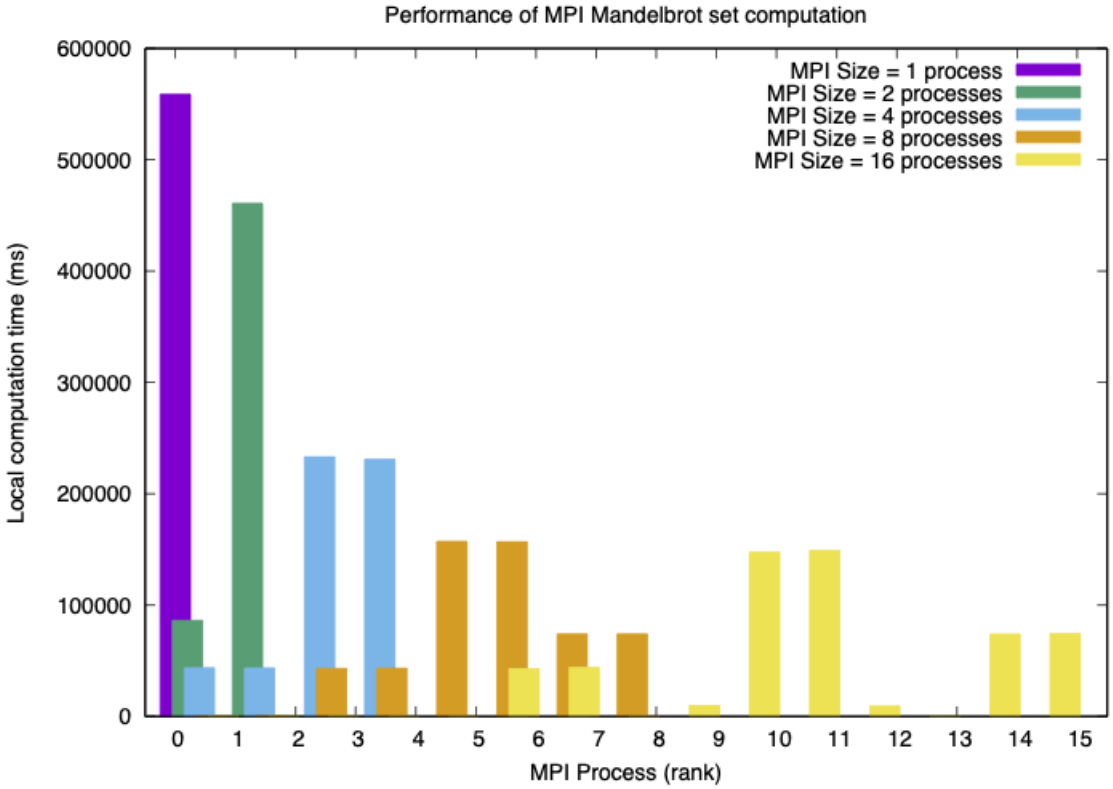
Using the given specifications the desired output was achieved.

```
[stud07@icsnode32 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
```

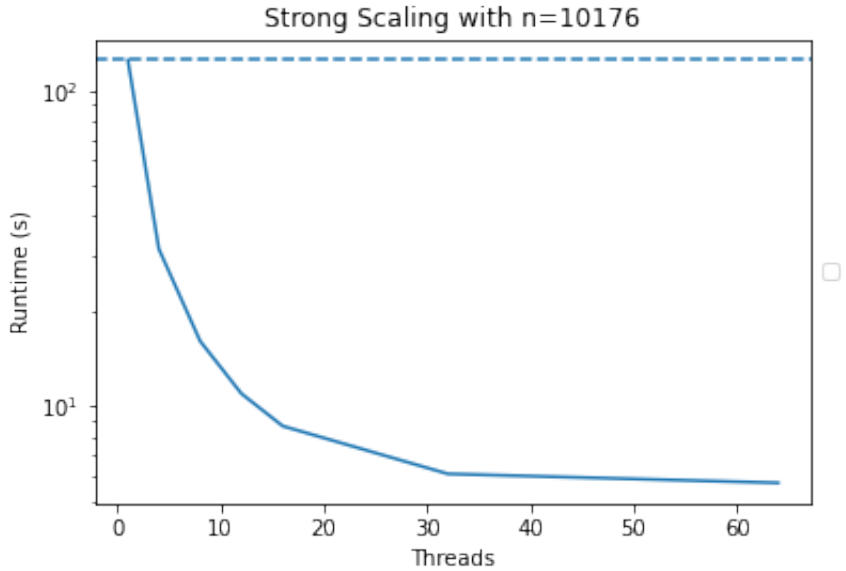
3. Parallelizing the Mandelbrot set using MPI [20 Points]

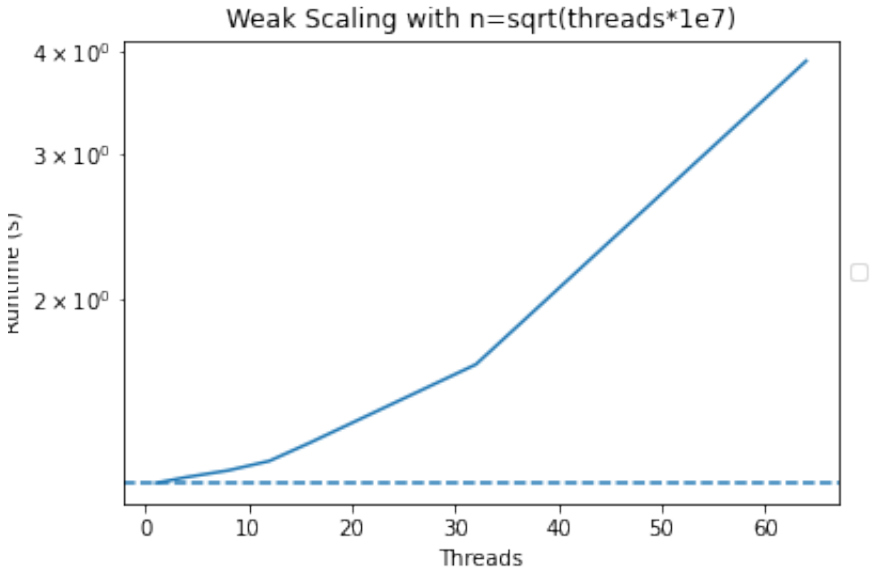
From the performance graph we can see that the average runtime per process decreases significantly. Since some processes seem to have more computation to do (perhaps due to more loops to converge in the mandelbrot computation) the runtime between n=8 and n=16 is not a hige difference. Because the total runtime is the runtime of the longest local runtime theyb are practically the same. Although we do see a major improvement form 1 to 2 to 4 to 8 processes, we may need to find a way to break up the computation so that it is more evenly distributed to achieve higher performance.





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




Weak Scaling with n=sqrt(threads*1e7)
4 × 10^0
3 × 10^0
2 × 10^0
Runtime (s)
0
10
20
30
40
50
60
Threads