

High-Performance Computing Lab

Student: Paolo Deidda

Institute of Computing

Discussed with: FULL NAME

Solution for Project 4

HPC Lab — Submission Instructions

(Please, notice that following instructions are mandatory:
submissions that don't comply with, won't be considered)

- Assignments must be submitted to iCorsi (i.e. in electronic format).
- Provide sources (e.g. C/C++ files, Matlab). If you are using libraries, please add them in the file. Sources must be organized in directories called:
Project_number_lastname_firstname
and the file must be called:
project_number_lastname_firstname.zip
project_number_lastname_firstname.pdf
- The TAs will grade your project by reviewing your project write-up, and looking at the implementation you attempted, and benchmarking your code's performance.
- You are allowed to discuss all questions with anyone you like; however: (i) your submission must list anyone you discussed problems with and (ii) you must write up your submission independently.

Contents

1. Task: Ring maximum using MPI [10 Points]	2
2. Task: Ghost cells exchange between neighboring processes [15 Points]	2
3. Task: Parallelizing the Mandelbrot set using MPI [20 Points]	2
4. Task: Parallel matrix-vector multiplication and the power method [40 Points]	2
5. Task: Quality of the Report [15 Points]	2

1. Task: Ring maximum using MPI [10 Points]

I determine each process message destination and source using its rank.

The communication can be implemented using the standard MPI functions `MPI_Send` and `MPI_Recv`, which take the following arguments:

- buffer address (to store sent or received data)
- number of elements (to send or receive)
- MPI datatype (of the elements to send or receive)
- destination (or source) rank (of the process to send to or receive from)
- message tag (to identify the message with an id)
- communicator (to identify the group of processes involved in the communication, useless in this case)

When using `MPI_Send` and `MPI_Recv` separately, the ranks must be split into two groups like even and odd ranks, otherwise all processes may block by attempting to send or receive at the same time.

Alternatively, we can use the `MPI_Sendrecv` function, which takes care of both sending and receiving in a single call, avoiding deadlocks. *dio negro*

```
21     MPI_Sendrecv(
22         &send_val, 1, MPI_INT, next, 0,
23         &recv_val, 1, MPI_INT, prev, 0,
24         MPI_COMM_WORLD, MPI_STATUS_IGNORE
25     );
```

Listing 1: Implementation from file ring_sum.c

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

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

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

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