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

High-Performance Computing Lab **Institute of Computing**

Student: Paolo Deidda 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]

I detrmine 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.

```
    MPI_Sendrecv(
      &send_val, 1, MPI_INT, next, 0,
      &recv_val, 1, MPI_INT, prev, 0,
      MPI_COMM_WORLD, MPI_STATUS_IGNORE
    );
```

Listing 1: Implemetation from file ring_sum.c

Once launched the job we can see the folloring output with `-ntasks=4` on the left and `-ntasks=8` on the right.

```
Process 0: Sum = 6
Process 2: Sum = 6
Process 1: Sum = 6
Process 3: Sum = 6
```

Listing 2: Output from file ring_57313.out

```
Process 7: Sum = 28
Process 5: Sum = 28
Process 6: Sum = 28
Process 3: Sum = 28
Process 0: Sum = 28
Process 1: Sum = 28
Process 2: Sum = 28
Process 4: Sum = 28
```

Listing 3: Output from file ring_57315.out

With flag `-ntasks=4` slurm automatically assign a node with at least 4 cores, since by default:

- 1 Slurm task = 1 MPI process
- 1 MPI process = 1 CPU core

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

I first create a 4 × 4 Cartesian communicator with periodic boundaries on both axes. Then, by using `MPI_Cart_shift`, I get the ranks of the neighboring processesnorth, south, east, and west, making sure the first and last ranks stay connected. To handle the column ghost layer, I create a derived datatype using `MPI_Type_vector`; this datatype covers one element for every `DOMAINSIZE` entries and goes all the way through the interior height of the tile.

I then keep using `MPI_Sendrecv` calls, one for each direction, to avoid any deadlock. Each call sends the interior boundary (the rows or columns that leave out the corner points) and directly receives the data into the ghost layer.

```
    //  to the top
    MPI_Sendrecv(&data[1 * DOMAINSIZE + 1], SUBDOMAIN, MPI_DOUBLE, rank_top, 0,
                 &data[0 * DOMAINSIZE + 1], SUBDOMAIN, MPI_DOUBLE, rank_top, 0,
                 comm_cart, &status);

    //  to the bottom
    MPI_Sendrecv(&data[(DOMAINSIZE - 2) * DOMAINSIZE + 1], SUBDOMAIN, MPI_DOUBLE,
                 rank_bottom, 1, &data[(DOMAINSIZE - 1) * DOMAINSIZE + 1],
                 SUBDOMAIN, MPI_DOUBLE, rank_bottom, 1, comm_cart, &status);
```

Listing 4: Two example from top and bottom ghost cells exchange from the file ghost.c with `SUBDOMAIN = 6`

Here the output of the program:

```
/****************************************************************
```

Listing 5: Two example from top and bottom ghost cells exchange from the file ghost.c with `SUBDOMAIN = 6`

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]**