Parallel and High Performance Computing

Dr. Pablo Antolín

Solution Series 6

April 3 2025

Advanced MPI

1 Pi

Exercise 1.1: *Derived types*

Check the solution in the file `pi_derived_datatype.cc` of the folder `pi/solution`. The creation of the datatype follows the example from the course's slides. You have to be careful to commit the datatype and free it at the end. Once you have your type you can use it in `MPI_Gather` by replacing `MPI_DOUBLE` with the name of your datatype (`sum_t`). For the broadcast the changes are similar.

Exercise 1.2: *Persistent communications*

Check the solution in the file `pi_p2p_persistent_ring.cc` of the folder `pi/solution`. No difficulties here, just declare the persistent communications before the loop and replace the asynchronous communications by calls `MPI_Start` or `MPI_Startall`. Notice that the `MPI_Wait` for the received must be called before the `MPI_Wait` for the send.

2 Write BMP

Exercise 2.1: ***MPI I/O*** In this exercise there is a problem in the code. The copy in the `img` buffer flips the y axis.

```
float v = ((m_grid(h - 1 - i, j) - m_min) / (m_max - m_min));
```

this should be

```
float v = ((m_grid(i, j) - m_min) / (m_max - m_min));
```

To write the file then you have to be careful to let the process 0 write the header and then all process can write there part offset by $54 + h \times$ `prank`. You also have to adapt the `filesize`.

3 Poisson stencil with mpi4py

Both implementations can be found in the scripts `poisson.py` and `simulation.py`. To run the code with blocking communications, pass as a parsing argument -sync “sync”, otherwise pass -async “async”.