MPI Identity Matrix

Nandhana Sakthivel

Exercise 4

Objective

The objective of this exercise is to initialize an **Identity Matrix** in **MPI**.

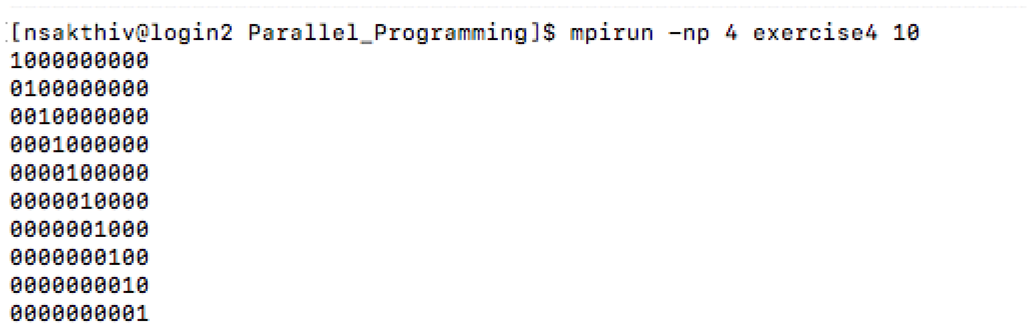
Result

We consider an **Identity Matrix** of size **(N,N)**. The matrix is initialised by distributing it among the number of processes mentioned. If the size of the matrix is $\mathbf{N < 10}$, it is printed otherwise the initialised matrix is written in a binary file.

Here we are using **Non-blocking Communication** and **Blocking Communication** to understand the difference between them. In non-blocking communication, **send start** call initiates the send operation, but does not complete it. The send start call will return before the message was copied out of the send buffer. A separate **send complete** call is needed to complete the communication, i.e., to verify that the data has been copied out of the send buffer. Similarly,a **receive start** call initiates the receive operation, but does not complete it. The call will return before a message is stored into the receive buffer. A separate **receive complete** call is needed to complete the receive operation and verify that the data has been received into the receive buffer.

$$MPI\_ISEND(buf, count, datatype, dest, tag, comm, request)$$

$$MPI\_IRECV(buf, count, datatype, source, tag, comm, request)$$



```
[nsakthiv@login2 Parallel_Programming]$ mpirun -np 4 exercise4 10
1000000000
0100000000
0010000000
0001000000
0000100000
0000010000
0000001000
0000000100
0000000010
0000000001
```

Figure 1: Matrix of size 10*10

In **Blocking Communication**, both send and receive operation are synchronized and only after the whole data is received, both send and receive start the next task.

$$MPI\_SEND(buf, count, datatype, dest, tag, comm)$$
$$MPI\_RECV(buf, count, datatype, source, tag, comm, status)$$

Conclusion

From the exercise, we see that **Blocking communication** is used commonly because it is easier to use and we don't need to synchronize the functions. We use **Non-Blocking communication** when we need to overlap computation and communication which is efficient, but we need to take care of the synchronization process.