Ring Communication:Blocking and Non Blocking

By: Rabindra Khadka

Overview:

The objective of this exercise was to implement a program that allows a processor to communicate its rank around a ring . The sum of all ranks was aggregated and then printed out by each processor.

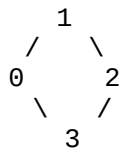

Fig 1: Schema of the ring communication structure among processors. Message is sent from 0 to 1 to 2 to 3 and back to 0 again.

The rank of each processor is stored in MPI_COMM_WORLD as an integer and this integer value is sent to the processor on its right . The same processor then receives from its left neighbor. The sum operation is performed on the received integer and its tracked .The values is passed in the ring until the processors gets their own value .At the end of the task, the sum of the values is printed out by each process.

The two programs under this lab exercise is carried out using two different APIs demonstrating blocking and non-blocking operations *MPI_Sendrecv()* and non blocking ,overlapping ,synchronous send i.e. *MPI_Issend()*.Out of the two , one of them calculates the sum of the single integer and other program performs the vector sum.

MPI_Sendrecv api combines in one call the sending of a message to one destination and receiving of another message from another process in a chain or ring fashion avoiding deadlock. This is a blocking send and receive operation while MPI_Isend is a non blocking operation.

Outcome and Procedure:

```
[rkhadka@login2 Parallel_Programming]$ mpirun -np 4 ./nb_sum.x 4
Proc 0 sum = 6
Proc 1 sum = 6
Proc 2 sum = 6
Proc 3 sum = 6
```

Fig 2: Showing the result after running the vector sum in 4 processors with a ring topology set up.

The procedure for compiling the code are as follows: (a) module load openmpi (b) mpicc -o exec file.c

(c) mpirun -np nprocs ./ exec N *## here N is the vector size for adding vector elements.*