Non-blocking Operations Application

Non-blocking Operations Application

Create a program with two processes. Process 0 will initialize the following array **message** of 8 positions:

0	1	2	3	4	5	6	7

Next, it will send this array to the Process 1 using a non-blocking operation. Then, it will compute the sum of the array **message**.

Non-blocking Operations Application

Process 1 will receive the **message** array using a non-blocking operation. It will then compute the sum of this array.

In the program show all the summations results for both processes. In addition, tell the user when the Process 0 has sent the array and when the Process 1 has received it.

What are the results? How can we perform this task by ensuring the synchronization between the processes and compute always the correct sum of the **message** array in the Process 1?

Non-blocking Operations

- int MPI_Isend(const void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm, MPI_Request *request);

- int MPI_Irecv(void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Request *request);

MPI_Isend

- Initiates the sending of a message but does not wait for the completion of the communication. It allows the program to continue with other computations or communication operations while the data is being sent in the background.

- int MPI_Isend(const void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm, MPI_Request *request);

MPI_Irecv

- Initiates the receiving of a message but does not wait for the completion of the communication. It allows the program to continue with other computations or communication operations while waiting for incoming data in the background.

- int MPI_Irecv(void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Request *request);