MA513

Parallel Computing

Assignment-2 Report

Paranjay Bagga
160101050

Problem Statement :-

Introduction to MPI Programming: Getting familiar with various communication modes along with the associated APIs.

System Specifications :-

All the experiments have been performed on **Dell Inspiron 5559** laptop with **Intel i5-6200U dual core processor** and **8 GB RAM**. Each core has **2 threads.**

Experiments :-

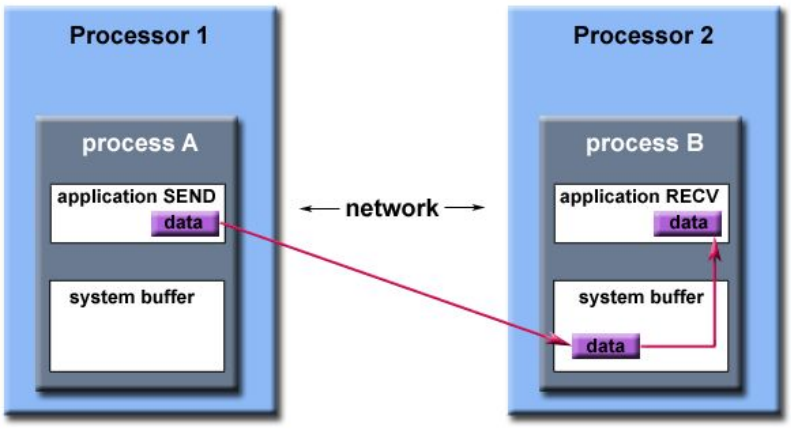
Experiments are done on programs in which the data is sent in a cyclic fashion, i.e. node_1 sends data to node_2 which in turn sends data to node_3 and so on till node_N sends data to node_1. Different MPI APIs are used for sending and receiving data that may be blocking or non-blocking as are discussed below.

Different Communication Modes :-

SEND	Blocking	Nonblocking
Standard	mpi_send	mpi_isend
Ready	mpi_rsend	mpi_irsend
Synchronous	mpi_ssend	mpi_issend
Buffered	mpi_bsend	mpi_ibsend

RECEIVE	Blocking	Nonblocking
Standard	mpi_recv	mpi_irecv

- **MPI_Send :** This standard mode is actually a "non-mode" that lets the MPI implementation choose which communication mode is preferable. In OpenMPI, the observed behaviour is that for short messages, the send is automatically buffered, and hence ***non-blocking*** while for long messages, the message will be sent using a mode somewhat close to the synchronous mode, and hence will be ***blocking***.
    - **Observation** : No deadlock is observed until the size of the user-defined buffer exceeds the value of ***16356*sizeof(int)***.



Path of a message buffered at the receiving process

- **MPI_Rsend :** This is a standard blocking send which will block until the recipient has received the message. The difference with MPI_Send is the ‘R’ which stands for ***ready***. This routine requires the recipient to have issued the corresponding receive routine (MPI_Recv or MPI_Irecv) before MPI_Rsend is invoked which may improve performance by saving the time normally spent hand-shaking with the receiving MPI process.
    - **Observation** : No deadlock is observed until the size of the user-defined buffer exceeds the value of ***16356*sizeof(int)***.

- **MPI_Bsend :** This is the asynchronous blocking send (the capital ‘B’ standing for ***buffered***). it will block until a copy of the buffer passed is made. As a blocking send, MPI_Bsend guarantees that the buffer passed can be safely reused once MPI_Bsend returns. Here, the size of the copy is equal to that of the buffer passed plus the memory overhead generated by an MPI_Bsend, represented by *MPI_BSEND_OVERHEAD*, which contains for instance the rank of the recipient process. The MPI buffer in which the copy will be made must be allocated by the user and explicitly attached to MPI using *MPI_Buffer_attach*.
    - **Observation** : For sending data of size **N*(sizeof(int))**, no deadlock is observed until the size of the buffer is at least ***MPI_BSEND_OVERHEAD + (N-2)*sizeof(int)***. (where MPI_BSEND_OVERHEAD = 96)

- **MPI_Ssend :** This is the synchronous blocking send (the capital ’S’ standing for ***synchronous***). it will block until the recipient has received the message. As a blocking send, MPI_Ssend guarantees that the buffer passed can be safely reused once MPI_Ssend returns. Also, this routine may be invoked implicitly by the standard blocking send (MPI_Send).
    - **Observation** : Deadlock is observed for all values of N.

- **MPI_Sendrecv :** This is a combination of an ***MPI_Send*** and an ***MPI_Recv***. It can be seen as having both subroutines executed concurrently. The difference with ***MPI_Sendrecv_replace*** is that with MPI_Sendrecv the buffers used for send and receive must be different. This allows for instance MPI_Sendrecv to send and receive different amounts and types of data.
    - **Observation** : No deadlock is observed for any value of N.

- **MPI_Isend :** This is the standard non-blocking send (the capital 'I' stands for ***immediate return***). MPI_Isend will make that decision itself; it will issue an asynchronous non-blocking send (***MPI_Ibsend***) if there is enough space in the buffer attached to MPI (*MPI_Buffer_attach*) to copy the buffer passed, issuing a synchronous non-blocking send (***MPI_Issend***) otherwise. Either way, as a non-blocking send, MPI_Isend will not block until the buffer passed is safe to be reused. ***MPI_Wait*** waits for a non-blocking operation to complete. Other non-blocking sends are MPI_Ibsend, MPI_Issend, ***MPI_Irsend***; with their non-blocking receiving counterpart being ***MPI_Irecv***.
    - **Observation** : No deadlock is observed for any value of N.