Computer Science & Information Systems

**Systems for Data Analytics - Lab Sheet 4**

**Message Passing Model – Send and Receive API**

1. Objective:

Students should be able to

1. Get familiarity with the Message Passing model in Parallel computing environment
2. Get hands-on experience on MPI APIs – MPI\_Send, MPI\_Recv

This lab sheet provides a quick introduction to using MPI(Message passing Interface). This exercise will introduce the send and receive API used in MPI in C language for Peer to Peer communication.

A *process* is (traditionally) a program counter and address space. Processes may have multiple *threads* (program counters and associated stacks) sharing a single address space.

MPI is for communication among processes, which have separate address spaces.

**Interprocess communication consists of** Synchronization and Movement of data from one process’s address space to another’s.

The message-passing approach makes the exchange of data *cooperative*. Data is explicitly *sent* by one process and *received* by another. An advantage is that any change in the receiving process’s memory is made with the receiver’s explicit participation. Communication and synchronization are combined.

Process 0

Process 1

**Send(data)**

**Receive(data)**

MPI’s blocking sending and receiving functions –

MPI\_Send(

void\* data,

int count,

MPI\_Datatype datatype,

int destination,

int tag,

MPI\_Comm communicator)

MPI\_Recv(

void\* data,

int count,

MPI\_Datatype datatype,

int source,

int tag,

MPI\_Comm communicator,

MPI\_Status\* status)

* The first argument is the data buffer.
* The second describe the count of elements that reside in the buffer.
* The third arguments describe the type of elements that reside in the buffer. MPI\_Send sends the exact count of elements, and MPI\_Recv will receive at most the count of elements.
* The fourth and fifth arguments specify the rank of the sending/receiving process and the tag of the message.
* The sixth argument specifies the communicator and the last argument (for MPI\_Recv only) provides information about the received message.

1. Steps to be performed:

Lab used – remote lab with Machines – MPI01 and MPI02

Language used – C

To compile: mpicc –o <output-file> <source-file>.c

To run: mpirun -n <min no. of processes> ./<output-file>

1. Outputs/Results:

mpirun -n 1 ./<output-file>

error

Or

mpirun -n 2 ./<output-file>

Process 1 received number -1 from process 0

**Make sure the executable is present in the same path in all the machines/ hosts with the same name.**

1. Observations:

* Students to observe the output when executing with one process. Justify your observation.
* Execute and observe if the no. of process is 2.
* Processes can be created and executed in multiple hosts in the same network. Try executing in two machines and observe the output.