GOVERNMENT COLLEGE OF ENGINEERING, JALGAON

(An Autonomous Institute of Government of Maharashtra)

Department of Computer Engineering

Name: Vaibhav Bharat Sontakke PRN No: 1942207

Class: T. Y. B. Tech. Batch: L4

Date of Performance: _____ Date of Completion: _____

Subject: DSL Sign. of Teacher with Date: _____

Aim: Design a distributed application using Message Passing Interface (MPI)

Title: Practical for a distributed application using Message Passing Interface (MPI) for remote computation where client submits a string to the server and server returns the reverse of it to the client.

Theory:

Message Passing Interface (MPI)

Objective:

Message Passing Interface (MPI) is a standardized and portable message- passing system distributed and developed for parallel computing. MPI provides parallel hardware vendors with a clearly defined base set of routines that can be efficiently implemented.

The message passing interface (MPI) is a standardized means of exchanging messages between multiple computers running a parallel program across distributed memory.

In parallel computing, multiple computers -- or even multiple processor cores within the same computer -- are called nodes. Each node in the parallel arrangement typically works on a portion of the overall computing problem. The challenge then is to synchronize the actions of each parallel node, exchange data between nodes and provide command and control over the entire parallel cluster. The message passing interface defines a standard suite of functions for these tasks.

FEATURES:

1. General
   - Communicators combine context and group for message security
   -Thread safety
2. Point-to-point communication
   -Structured buffers and derived datatypes, heterogeneity
   - Modes: normal (blocking and non-blocking), synchronous, ready (to allow access to

fast protocols), buffered

3. Collective

- Both built-in and user-defined collective operations
- Large number of data movement routines - Subgroups defined directly or by topology

4. Non-message-passing concepts not included:

· process management

· remote memory transfers

· active messages

· threads

· virtual shared memory

5. MPI does not address these issues, but has tried to remain compatible with these ideas (e.g. thread safety as a goal, intercommunicators)

Working of MPI in Distributed Computing:

· **Distriuted Memory**

Every processor has its own local memory which can be accessed directly only by its own CPU. Transfer of data from one processor to another is performed over a network. Differs from shared memory systems which permit multiple processors to directly access the same memory resource via a memory bus.

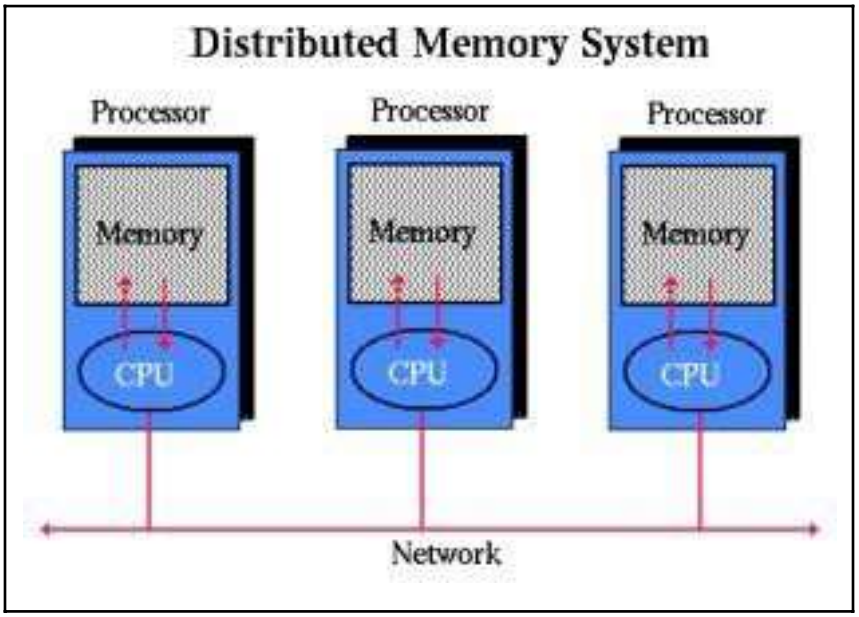


Figure: Message Passing Interface in DS

· **Message Passing**

The method by which data from one processor's memory is copied to the memory of another processor. In distributed memory systems, data is generally sent as packets of information over a network from one processor to another. A message may consist of one or more packets, and usually includes routing and/or other control information.

· **Processor**

A process is a set of executable instructions (program) which runs on a processor. One or more processes may execute on a processor. In a message passing system, all processes communicate with each other by sending messages - even if they are running on the same processor. For reasons of efficiency, however, message passing systems generally associate only one process per processor.

· **Message Passing Library**

Usually refers to a collection of routines which are imbedded in application code to accomplish send, receive and other message passing operations.

· **Send / Receive**

Message passing involves the transfer of data from one process (send) to another process (receive). Requires the cooperation of both the sending and receiving process. Send operations usually require the sending process to specify the data's location, size, type and the destination. Receive operations should match a corresponding send operation.

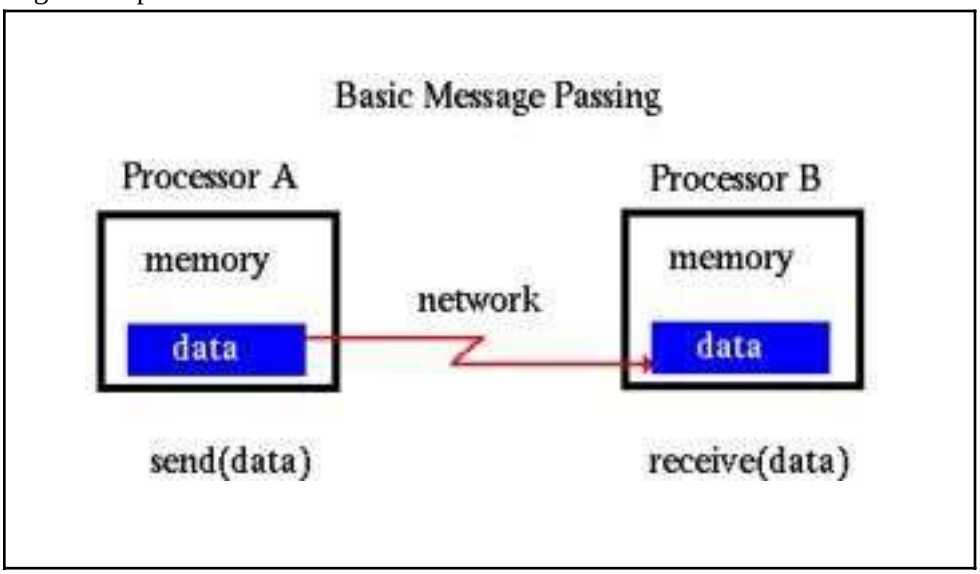


Figure: Send & Receive Msg in MPI in Network

· **Synchronous / Asynchronous**

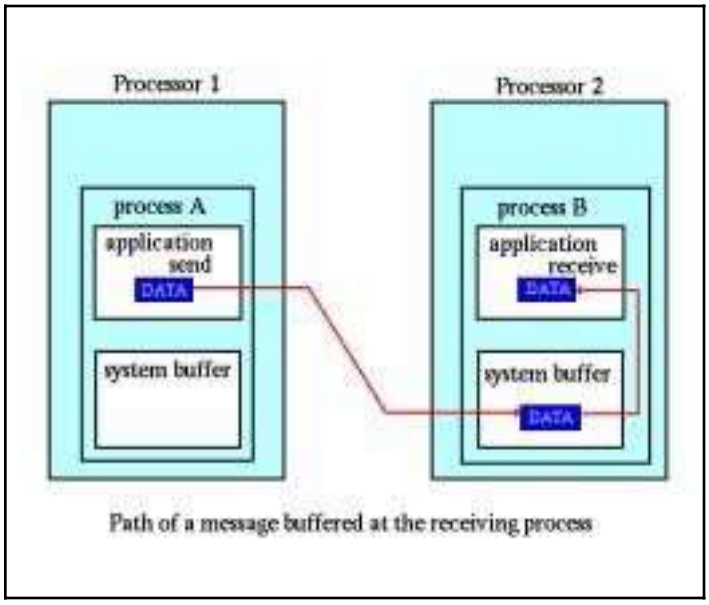
A synchronous send operation will complete only after acknowledgement that the message was safely received by the receiving process. Asynchronous send operations may "complete" even though the receiving process has not actually received the message.

· **Application Buffer**

The address space that holds the data which is to be sent or received. For example, your program uses a variable called, "in msg". The application buffer for in msg is the program memory location where the value of in msg resides.

· **System Buffer**

System space for storing messages. Depending upon the type of send/ receive operation, data in the application buffer may be required to be copied to/from system buffer space. Allows communication to be asynchronous.



· **Blocking Communication**

A communication routine is blocking if the completion of the call is dependent on certain "events". For sends, the data must be successfully sent or safely copied to system buffer space so that the application buffer that contained the data is available for reuse. For receives, the data must be safely stored in the receive buffer so that it is ready for use.

· **Non-blocking Communication**

A communication routine is non-blocking if the call returns without waiting for any communications events to complete (such as copying of message from user memory to system memory or arrival of message).

It is not safe to modify or use the application buffer after completion of a non-blocking send. It is the programmer's responsibility to insure that the application buffer is free for reuse.

Non-blocking communications are primarily used to overlap computation with communication to effect performance gains.

Advantage of MPI

1. Standardization: MPI is the only message passing library which canbe considered a standard.
2. Portability: no need to modify your source code when you portyour application to a different platform.
3. Functionality: Many routines available to use.
4. Availability: A variety of implementations are available. **Disadvantages of MPI:**

1. Lack of scalability between memory and CPU. Adding more CPUs can increase the traffic on shared memory associated with cache andmemory management.

2. Programmer responsibility for synchronization construct that ensure correct access of global memory.

3. Expense: It becomes increasingly difficult and expensive to design and produce shared memory machines with ever increasing number of processors. **Steps to Implement Interface**

Server site:

1. # open terminalsudo apt-get update

sudo apt-get install libopenmpi-dev

mkdir exp3

cd exp3 gedit

server.c

2. pass the variablesMPI_Comm client;

   MPI_Status status;

   char port_name[MPI_MAX_PORT_NAME],

   str[50],ch,temp;

   int size, again, i,j;
3. MPI_Open_port(MPI_INFO_NULL, port_name);

   printf("Server available at port: %s\n", port_name);
4. send the string at server site
5. send the reversed string to client (character by character) ,end the string(tag=1)
6. MPI_Comm_disconnect(&client);
7. # save and exit the file

Client site:

8. gedit client.c
9. accept input string
10. Receive the reversed string from server and display it
11. printf("\nReversed string is : %s\n\n",str);
12. MPI_Comm_disconnect(&server);
13. # save and exit the file
14. Compile and run

Conclusion: Hence We Sucessfully Study a distributed application using Message Passing Interface (MPI)

……...……

…….....

Miss. Archana Chitte

Name & Sign of Course Teacher