# HPC unit 3

MPI (message passing interface) : a standardised and portable message passing system designed to function on a wide variety of parallel computing architectures. It enables the development of parallel programs that can run efficiently on both small clusters & large supercomputers.

It's a protocol used for programming parallel computers. Allows processes to communicate with one another by sending & recieving msgs.

MPI program structure:

MPI include file

Program begins

↓

Initialise MPI environment (Parallel code)

↓

Do work & make msg passing calls

↓

Terminate MPI env.

⋮

Program ends.

MPI Communicators and Groups:

It uses objects called communicators & groups to define which collectn of processes may communicate with each other.

MPI_COMM_WORID : default communicator that includes all the processes initiated by the MPI program. Every process within MPI_Comm_World can communicate with any other process in the same communicator.

Level of thread support:

MPI_thread_single : level 0: only one thread will execute.

MPI_thread_funneled (level 1): multiple threads may exist, but only main thread makes mpi calls.

MPI_thread_serialized (level 2): multiple threads can make mpi calls, but not concurrently; calls are serialized.

MPI_thread_multiple (level 3): multiple threads can make mpi calls without restrictions.

1) MPI_Init : Initialize MPI execut^n environment

Synopsis :

int MPI_Init(int *argc, char ***argv)

I/p params:

argc : pointer to no. of args.

argv : pointer to the arg vector.

must be called by one thread only, that thread is called main thread & must be the thread that calls MPI_finalise.

2) MPI_Comm_Size : determines the size of grp associated with communicator.

int MPI_Comm_Size (in MPI_Comm comm, int *size)

I/P : communicator

O/P : no. of processes in the grp of comm int.

3) MPI_Comm_rank : determines the rank of calling processes in communicator

Synopsis: int MPI_Comm_rank (MPI_Comm comm, int * rank)

I/P : Comm
O/p : rank

4) MPI_Send : sends a msg to specified destination

I/P params :
1) buf : initial address of send buffer
2) count : no. of elements in " "
3) datatype : datatype of each send buffer elem
4) dest : rank of dest^n process
5) tag : message tag
6) comm : comm handle.

5) MPI_Recv : recieves msg from specified src.

I/P : same as mpi-send except buf.
O/p : buf : initial address of recieve buffer
status : status of recieved msg.

MPI_Scatter : distributes distinct chunks of data from the root process to all processes in a communicator.

Syntax:
int MPI_Scatter (const void *sendbuf, int send count, MPI_Datatype sendtype, void * recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)

MPI_Gather : collects chunks of data from all processes and gathers them into a single buffer on root process.

Syntax same as above.

MPI_Bcast : broadcast data from the root process to all other processes in communicator

MPI_Barrier : blocks processes until all have reached this routine, synchronizing them