ASSIGNMENT No.: 3

Problem Statement:

Write a MPI Program for calculating a quantity called coverage from data files

Objective:-

1.To understand concept of **Message Passing Interface(MPI)**
2. To effectively use multi-core or distributed, concurrent/Parallel environments.
3. To develop problem solving abilities using Mathematical Modeling

Theory:

Concept of MPI:

MPI is a ***specification*** for the developers and users of message passing libraries. By itself, it is NOT a library - but rather the specification of what such a library should be.

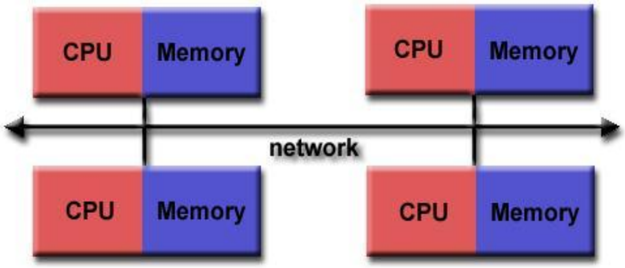
MPI primarily addresses the message-passing parallel programming model: data is moved from the address space of one process to that of another process through cooperative operations on each process.

Simply stated, the goal of the Message Passing Interface is to provide a widely used standard for writing message passing programs.

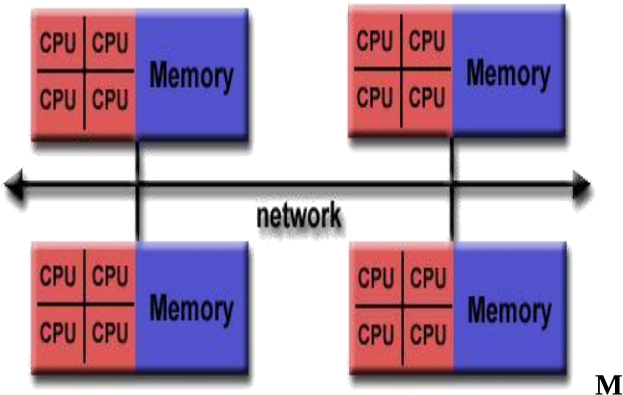
The Message Passing Interface Standard (MPI) is a message passing library standard based on the consensus of the MPI Forum, which has over 40 participating organizations, including vendors, researchers, software library developers, and users.

The goal of the Message Passing Interface is to establish a portable, efficient, and flexible standard for message passing that will be widely used for writing message passing programs. MPI is not an IEEE or ISO standard, but has in fact, become the "industry standard" for writing message passing programs on HPC platforms.

MPI Programming Model:

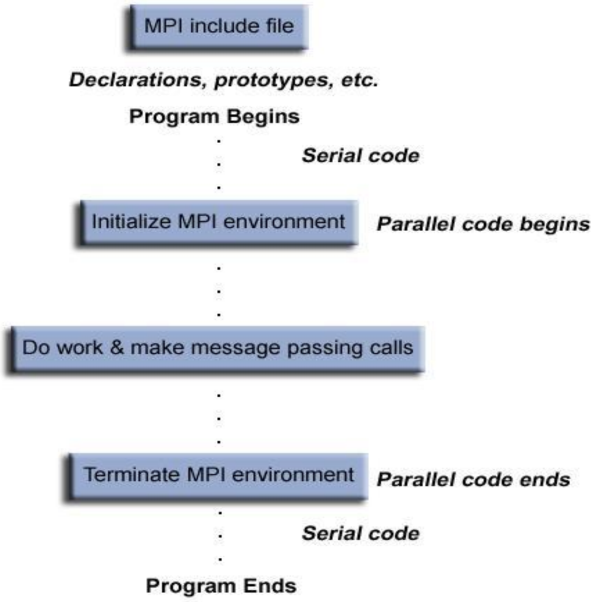


Simple P4 machines with single core



Multicore machines like dual core, i3,i5,i7

General MPI Program Structure

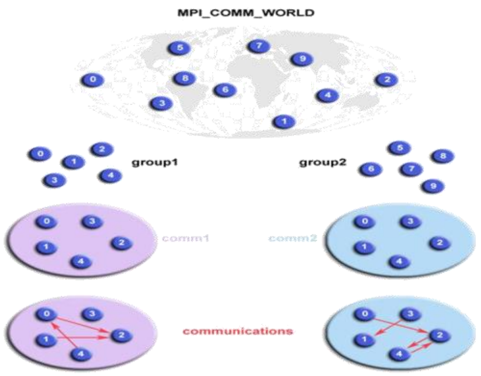

MPI include file
Declarations, prototypes, etc.
Program Begins
Serial code
Initialize MPI environment
Parallel code begins
Do work & make message passing calls
Terminate MPI environment
Parallel code ends
Serial code
Program Ends

Communicators and Groups:

MPI uses objects called communicators and groups to define which collection of processes may communicate with each other.

Most MPI routines require you to specify a communicator as an argument.

MPI_COMM_WORLD whenever a communicator is required - it is the predefined communicator that includes all of your MPI processes.



Level of Thread Support:
MPI libraries vary in their level of thread support:

- MPI_THREAD_SINGLE - Level 0: Only one thread will execute.
- MPI_THREAD_FUNNELED - Level 1: The process may be multi-threaded, but only the main thread will make MPI calls - all MPI calls are funneled to the main thread.
- MPI_THREAD_SERIALIZED - Level 2: The process may be multi-threaded, and multiple threads may make MPI calls, but only one at a time. That is, calls are not made concurrently from two distinct threads as all MPI calls are serialized.
- MPI_THREAD_MULTIPLE - Level 3: Multiple threads may call MPI with no restrictions.

Pros of MPI:

- runs on either shared or distributed memory architectures
- can be used on a wider range of problems than OpenMP
- each process has its own local variables
- distributed memory computers are less expensive than large shared memory computers

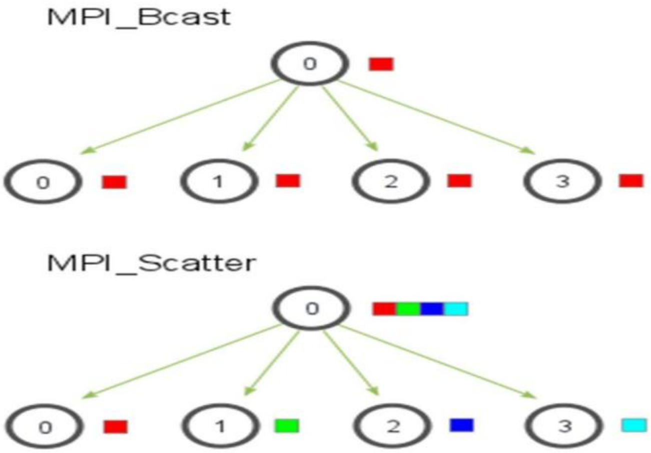
Cons of MPI:

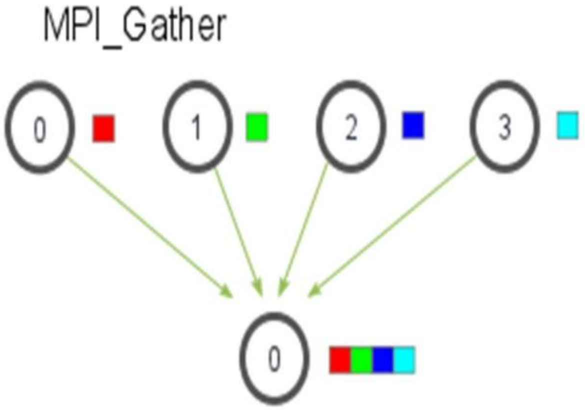
O requires more programming changes to go from serial to parallel
version O can be harder to debug
O performance is limited by the communcation network between the nodes

MPI Scatter, Gather:

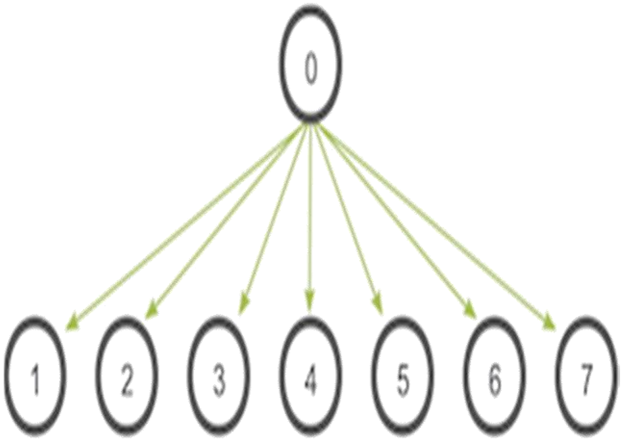
```
MPI_Gather(
   void* send_data,
   int send_count,
   MPI_Datatype send_datatype,
   void* recv_data,
   int recv_count,
```

MPI_Datatype recv_datatype,
int root,
MPI_Comm communicator)





MPI_Bcast(
void* data,
int count,
MPI_Datatype datatype,
int root,
MPI_Comm communicator)



Mathematical Model:

Let ,

S = { s, e, x, y, Fm, Si, DD, NDD }
s = Initial State, i.e. MPI_Init()
e = End State , i.e.MPI_Finalize()
Si = Intermediate states
x = Input values i.e. Numbers in random
manner y = Output/Result. i.e. Sum of numbers
Fm=Main function or algorithm that gives specific output i.e. MPI_Scatter(), MPI_Gather().
NDD = Non deterministic data
DD = Deterministic data

Positive Test Cases:

Sr. No.	Test Condition	Steps to be executed	Expected Result	Actual Result
1.	Enter the number of process	Press Enter	Assign the numbers to n process	Same as Expected

Negative Test Cases:

Sr. No.	Test Condition	Steps to be executed	Expected Result	Actual Result
1.	Entered input data is in character or symbol	Press enter	Error message	Display result

Conclusion:

Thus, we have implemented a MPI Program for calculating a quantity called coverage from data files.