Assignment - 2

Title :-
To develop any distributed application using Message Passing Interface (MPI)

Objective :-
By the end of this assignment, the student will be able to implement any distributed applications based on MPI.

Theory :-

Message Passing Interface :-

- Message passing is a popularly renowned mechanism to implement parallelism in applications. It is also called as MPI.

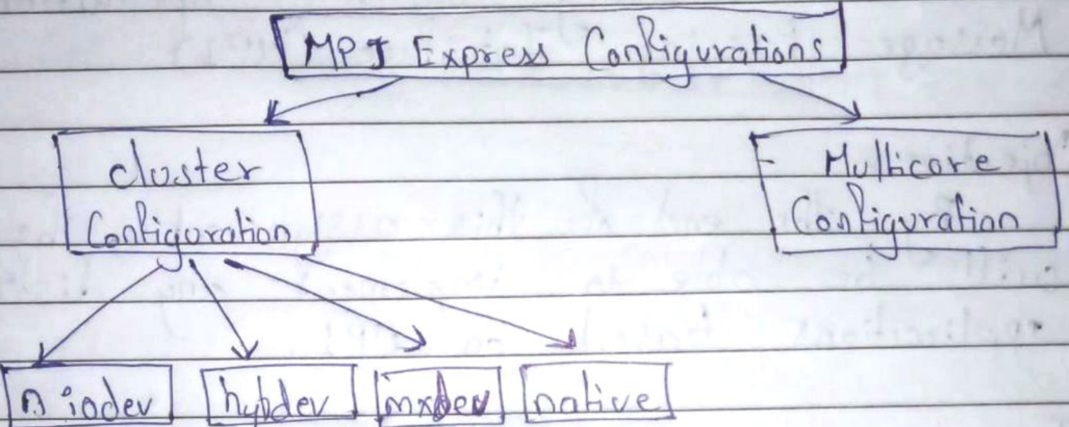
- MPJ is a message passing library that can be used by application developers to execute their parallel Java applications on compute clusters or network of computers.

- MPJ is familiar Java API for MPI implementation

- The programming model followed by MPJ Express is Single Program Multiple Data (SPMD)

MPJ : MPJ with Java

MPJ Express Configuration:



- MPJ Express Software can be configured in two ways as shown in figure.

- Multicore Configur
  It is used to execute MPJ express user programs on laptops & desktops.

- Cluster Configur
  It is used to execute MPJ express user programs on clusters or network of computers.

* Installing MPJ Express:-

1. Download MPJ express (mpj.jar) & unpack it.

2. Set environment variables MPJ_HOME and PATH:

   - export MPJ_HOME = /path/to/mpj/
   - export PATH = $MPJ_HOME/bin: $PATH

3. Create a new ~~dir~~ working directory for MPJ express programs.
   eg:- /mpj-user directory.

4. Compile the MPJ Express library: cd $MPJ_HOME; ant

MPI Environment:-

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

- Group is the set of processes that communicate with one another.

- Communicator is the central object for communication in MPI

• There is default communicator whose group contains all initial~~s~~ processes, called MPI_COMM_WORLD.

• Every MPI program must contain import mpi.MPI

• MPI_Init initializes the execution environment for MPI.

• A process is identified by its rank in the group associated with communicator.

• How many processes are participating in this computation?

1. MPI_Comm_size function reports the number of processes.

2. MPI_Comm-rank function reports the rank, a number between 0 and size-1, identifying the calling process.

• MPI_Finalize cleans up all the extraneous mess that was first put into place by MPI_Init.

* Steps for Compilation & execution.

• Installing MPJ Express programs in the Multicore configuration.

1. Download MPJ express (mpj.jar) & unpack it.

2. Set MPJ_HOME and PATH environment variables

```
export MPJ_HOME = /path/to/mpj/
export PATH = $MPJ_HOME/bin: $PATH
```

Add these lines to ~/.bashrc

3. Compile :-

```
javac -cp $MPJ_HOME/lib/mpj.jar
      program.java
```

• Execute :-

```
$MPJ-HOME /bin/ mpjrun.sh -np 4 Program.
```

Conclusion:
In this assignment, I installed MPJ Express & created a program for sending & recieving prime numbers using MPI.