# Veermata Jijabai Technological Institute, Mumbai 400019

**Experiment No.:** 07

**Aim :** Implementation using MPI .

i. Calculating Rank and Number of processors.,
ii. Pi calculation.,
iii. Advanced MPI program that has a total number of 4 processes, where the process with rank = 0 should send VJTI letter to all the processes using MPI_Scatter call.,
iv. Find the maximum value in array of six integers with 6 processes, and print the result in root process using MPI_Reduce call,
v. Ring topology.

**Name:** Kiran K Patil

**Enrolment No.:** 211070904

**Branch:** Computer Engineering

**Batch:** IV

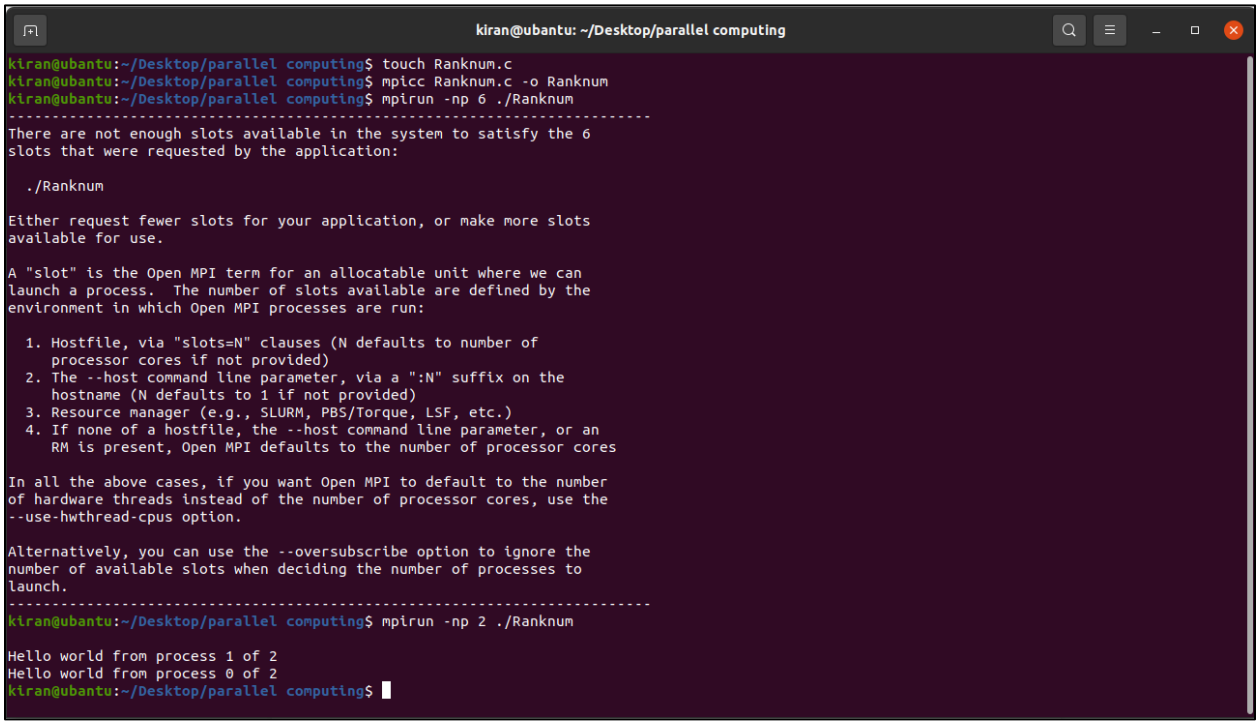
## 1. Calculating Rank and Number of processors.,

Objective - To write a simple MPI program for calculating Rank and Number of processor.

**Program :**

```
#include <mpi.h>
#include <stdio.h>
int main (int argc, char* argv[]){
int rank, size;
MPI_Init (&argc, &argv); /* starts MPI */
MPI_Comm_rank (MPI_COMM_WORLD, &rank); /* get current process id */
MPI_Comm_size (MPI_COMM_WORLD, &size); /* get number of processes */
printf( "Hello world from process %d of %d\n", rank, size );
MPI_Finalize();
return 0;
}
```

**Output:**




**Conclusion:** Thus, I have implemented MPI program for calculating rank and number of processors.

## 2. Pi calculation

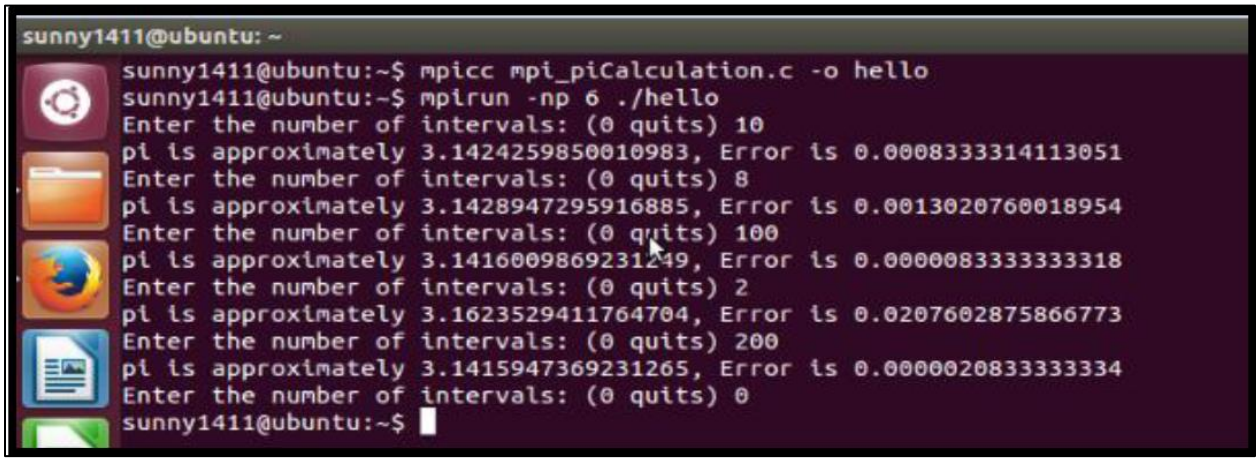
Objective - To write an MPI program for Pi calculation.

Program :

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>
int main( int argc, char *argv[] )
{
int n, myid, numprocs, I;
double PI25DT = 3.141592653589793238462643;
double mypi, pi, h, sum, x;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&myid);
while (1) {
if (myid == 0) {
printf("Enter the number of intervals: (0 quits) ");
scanf("%d",&n);
}
MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
if (n == 0)
break;
else {
h = 1.0 / (double) n;
sum = 0.0;
for (I = myid + 1; I <= n; I += numprocs) {
x = h * ((double)I - 0.5);
sum += (4.0 / (1.0 + x*x));
}
mypi = h * sum;

MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
MPI_COMM_WORLD);
if (myid == 0)
printf("pi is approximately %.16f, Error is %.16f\n",
pi, fabs(pi - PI25DT));
}
}
MPI_Finalize();
return 0;
}
```

Output :




**Conclusion:** Thus, I have implemented an MPI program for calculating the value of Pi.

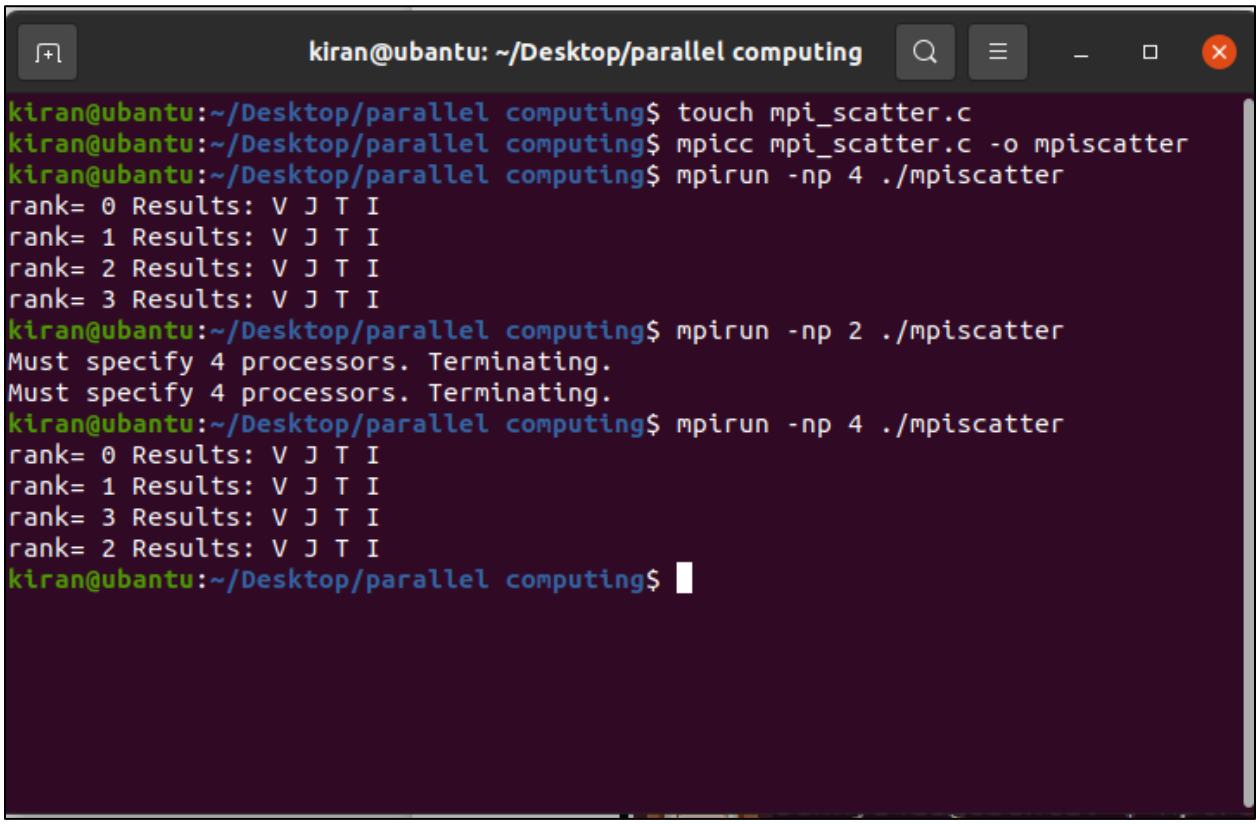
3. Advanced MPI program that has a total number of 4 processes, where the process with rank = 0 should send VJTI letter to all the processes using MPI_Scatter call.,

Program :

```
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4
int main (int argc, char *argv[]){
int numtasks, rank, sendcount, recvcount, source;
char sendbuf[SIZE][SIZE] = {
{'V','J','T','I'},
{'V','J','T','I'},
{'V','J','T','I'},
{'V','J','T','I'}};
char recvbuf[SIZE];
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
if (numtasks == SIZE) {
source = 0;
sendcount = SIZE;
recvcount = SIZE;
```

```
MPI_Scatter(sendbuf,sendcount,MPI_CHAR,recvbuf,recvcount,
MPI_CHAR,source,MPI_COMM_WORLD);
printf("rank= %d Results: %c %c %c %c\n",rank,recvbuf[0],
recvbuf[1],recvbuf[2],recvbuf[3]); }
else
printf("Must specify %d processors. Terminating.\n",SIZE);
MPI_Finalize();
}
```

Output :



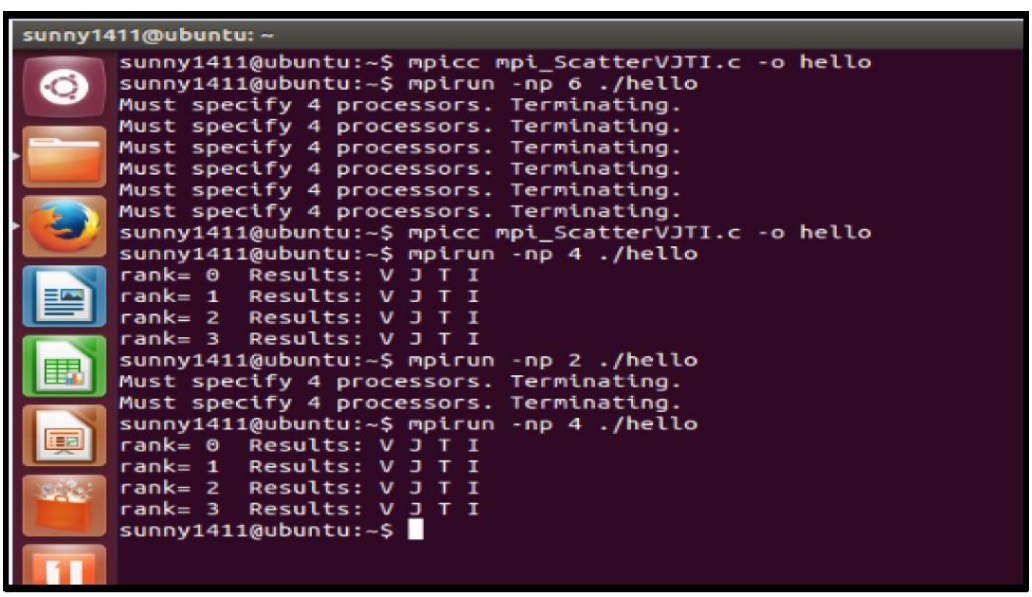

**Conclusion:** Thus, I have implemented an advanced MPI program for scattering “VJTI” to all the processes from the root process using MPI_Scatter Call.

4. Find the maximum value in array of six integers with 6 processes, and print the result in root process using MPI_Reduce call,

Program :

```
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4
int main (int argc, char *argv[])
{
int rank,numtasks,array[6] = {100,600,300,800,250,720},i,inputNumber;
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
printf("Local Input for process %d is %d\n",rank,array[rank]);
inputNumber = array[rank];
int maxNumber;
MPI_Reduce(&inputNumber, &maxNumber, 1, MPI_INT, MPI_MAX, 0,
MPI_COMM_WORLD);
// Print the result
if (rank == 0) {
printf("Maximum of all is: %d\n",maxNumber);
}
MPI_Finalize();
}
```

Output:



## 5. Ring topology.

To write an MPI program for Ring topology.

Program :

```
#include <stdio.h>
#include "mpi.h"
int main(int argc,char *argv[])
{
int MyRank, Numprocs, Root = 0;
int value, sum = 0;
int Source, Source_tag;
int Destination, Destination_tag;
MPI_Status status;
/* Initialize MPI */
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&Numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&MyRank);
if (MyRank == Root){
Destination = MyRank + 1;
Destination_tag = 0;
MPI_Send(&MyRank, 1, MPI_INT, Destination, Destination_tag,
MPI_COMM_WORLD);
}
else{
if(MyRank<Numprocs-1){
Source = MyRank - 1;
Source_tag = 0;
MPI_Recv(&value, 1, MPI_INT, Source, Source_tag,
MPI_COMM_WORLD, &status);
sum = MyRank + value;
Destination = MyRank + 1;

Destination_tag = 0;
MPI_Send(&sum, 1, MPI_INT, Destination, Destination_tag,
MPI_COMM_WORLD);
}
else{
Source = MyRank - 1;
Source_tag = 0;
MPI_Recv(&value, 1, MPI_INT, Source, Source_tag,
MPI_COMM_WORLD, &status);
sum = MyRank + value;
```

```
}
}
if (MyRank == Root)
{
Source = Numprocs - 1;
Source_tag = 0;
MPI_Recv(&sum, 1, MPI_INT, Source, Source_tag,
MPI_COMM_WORLD, &status);
printf("MyRank %d Final SUM %d\n", MyRank, sum);
}
if(MyRank == (Numprocs - 1)){
Destination = 0;
Destination_tag = 0;
MPI_Send(&sum, 1, MPI_INT, Destination, Destination_tag,
MPI_COMM_WORLD);
}
MPI_Finalize();
}
```

Output :


**Conclusion:** Thus, I have implemented an MPI program for ring topology.

## Conclusion:

- In conclusion, MPI (Message Passing Interface) is a popular programming model used for developing parallel applications that can run on a distributed computing system. It provides a standardized way of sending and receiving messages between processes, allowing for efficient communication and coordination between them.

- MPI programming can be complex and requires careful consideration of issues such as load balancing, synchronization, and data partitioning. However, it can significantly improve the performance of large-scale scientific simulations, data analytics, and other computationally intensive applications.