NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL
DEPARTMENT OF INFORMATION TECHNOLOGY
IT301 : Parallel Computing Lab
PC Lab 8 [Total marks = 10]
Date 05th September 2021

MPI Programming

Program 1. MPI non blocking Send and Receive(). Record the observation with and without MPI_Wait(). [2 Marks]
a) Note down results by commenting MPI_Wait()
b) Note down result by uncommenting MPI_Wait()
c) Note down the result by having mismatched tag .
In each case observe whether the process was waiting for completing send/recv or continuing its execution.

```
#include<mpi.h>
#include<stdio.h>
int main(int argc,char *argv[ ])
{
int size,myrank,x,i;
MPI_Status status;
MPI_Request request;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&size);
MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
if(myrank==0)
{
x=10;
printf("Process %d of %d, Value of x is %d sending the value x\n",myrank,size,x);
MPI_Isend(&x,1,MPI_INT,1,20,MPI_COMM_WORLD,&request);  // tag is different at receiver
//MPI_Wait(&request, &status);
}
else if(myrank==1)
{
printf("Value of x is : %d before receive\n",x);
MPI_Irecv(&x,1,MPI_INT,0,20,MPI_COMM_WORLD,&request);
printf("Receive returned immediately\n");
//MPI_Wait(&request, &status);
}

if (myrank==1) printf("Value of x is : %d after receive\n",x);
MPI_Finalize();
return 0;
}
```

//executing the program
//mpicc mpi_Isendrecv.c -o outp
//mpiexec -n 2 ./outp

Program 2: Demonstration of Bcast() [1 mark]

```
#include<mpi.h>
#include<stdio.h>
int main(int argc,char *argv[ ])
{
int size,myrank,x;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&size);
MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
printf("Before boradcast :Value of x in process %d : %d\n",myrank,x);
if(myrank==0){
scanf("%d",&x);
}
MPI_Bcast(&x,1,MPI_INT,0,MPI_COMM_WORLD);
printf("After Broadcast: Value of x in process %d : %d\n",myrank,x);
MPI_Finalize();
return 0;
}
```

Program 3: Demonstration of Reduce();
Note down the observation and explain the result. [2marks]

```
#include<mpi.h>
#include<stdio.h>
int main(int argc,char *argv[ ])
{
int size,myrank,i,x,y;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&size);
MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
x=myrank; // Note the value of x in each process.
MPI_Reduce(&x,&y,1,MPI_INT,MPI_SUM,0,MPI_COMM_WORLD);
if(myrank==0)
{
printf("Value of y after reduce : %d\n",y);
}
MPI_Finalize();
return 0;
}
```

Program 4: Demonstration of MPI_Gather();
Note down the observation and explain the result. [2marks]

```
#include<mpi.h>
#include<stdio.h>
int main(int argc,char *argv[ ])
{
int size,myrank,x=10,y[5],i;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&size);
MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
```

```
MPI_Gather(&x,1,MPI_INT,y,1,MPI_INT,0,MPI_COMM_WORLD); // Value of x at each process is copied to array y in Process 0
if(myrank==0)
{
for(i=0;i<size;i++)
printf("\nValue of y[%d] in process %d : %d\n",i,myrank,y[i]);
}
MPI_Finalize();
return 0;
}
```

Program 5: Demonstration of MPI_Scatter();
Note: Program is hardcoded to work with 4 processes receiving two chunks from array .
[2 mark]

```
#include<mpi.h>
#include<stdio.h>
int main(int argc,char *argv[ ])
{
int size,myrank,x[8],y[3],i;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&size);
MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
if(myrank==0)
{
printf("Enter 8 values into array x:\n");
for(i=0;i<8;i++)
scanf("%d",&x[i]);
}
MPI_Scatter(x,2,MPI_INT,y,2,MPI_INT,0,MPI_COMM_WORLD);
for(i=0;i<2;i++)
printf("\nValue of y in process %d : %d\n",myrank,y[i]);
MPI_Finalize();
return 0;
}
```

Program 6: Demonstration of MPI_Scatter() with partial scatter;
Note: Program is hardcoded to work with 3 processes receiving three chunks form array .
Notedown the difference between program 5 and program 6. [1 Mark]

```
#include<mpi.h>
#include<stdio.h>
int main(int argc,char *argv[ ])
{
int size,myrank,x[10],y[3],i;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&size);
MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
if(myrank==0)
{
printf("Enter 10 values into array x:\n");
for(i=0;i<10;i++)
```

```
scanf("%d",&x[i]);
}
MPI_Scatter(x,3,MPI_INT,y,3,MPI_INT,0,MPI_COMM_WORLD);
for(i=0;i<3;i++)
printf("\nValue of y in process %d : %d\n",myrank,y[i]);
printf("\nValue of y in process %d : %d\n",myrank,y[3]);
MPI_Finalize();
return 0;
}
```