

ROHIT RAJ

1RV17CS125

7<sup>TH</sup> CSE C2, PADP

PROGRAM 9

**CODE:**

```
# include <mpi.h>
```

```
# include <stdlib.h>
```

```
# include <stdio.h>
```

```
int main ( int argc, char *argv[] );
```

```
void ring_io ( int p, int id );
```

```
int main ( int argc, char *argv[] )
```

```
{
```

```
int error;
```

```
int id;
```

```
int p;
```

```
/* Initialize MPI.*/
```

```
MPI_Init( &argc, &argv );
```

```
/*Get the number of processes.*/
```

```
MPI_Comm_size( MPI_COMM_WORLD, &p );
```

```
/*Get the individual process ID.*/
```

```
MPI_Comm_rank( MPI_COMM_WORLD, &id );
```

```
/*Print a message.*/
```

```
if ( id == 0 )
```

```
{
```

```
printf ( "\n" );
```

```
printf ( "RING_MPI:\n" );
```

```
printf ( " C/MPI version\n" );
```

```
printf ( " Measure time required to transmit data around\n" );
```

```
printf ( " a ring of processes\n" );  
printf ( "\n" );  
printf ( " The number of processes is %d\n", p );  
}
```

```
ring_io( p, id );  
/*Shut down MPI.*/  
MPI_Finalize ( );  
/*Terminate.*/  
if ( id == 0 )  
{  
printf ( "\n" );  
printf ( "RING_MPI:\n" );  
printf ( " Normal end of execution.\n" );  
}  
return 0;  
}
```

```
void ring_io ( int p, int id )  
  
{  
int dest;  
int i;  
int j;  
int n;  
int n_test[5] = { 100, 1000, 10000, 100000, 1000000 };  
int n_test_num = 5;  
int source;  
MPI_Status status;  
double tave;  
int test;  
int test_num = 10;
```

```

double tmax;

double tmin;

double wtime;

double *x;


if ( id == 0 )
{
printf ( "\n" );

printf ( " Timings based on %d experiments\n", test_num );

printf ( " N double precision values were sent\n" );

printf ( " in a ring transmission starting and ending at process 0\n" );

printf ( " and using a total of %d processes.\n", p );

printf ( "\n" );

printf ( "      N      T min      T ave      T max\n" );

printf ( "\n" );

}

/*Choose message size.*/

for ( i = 0; i <n_test_num; i++ )
{
    n = n_test[i];


    x = ( double * ) malloc ( n * sizeof ( double ) );

    /*Process 0 sends very first message, then waits to receive the "echo" that has gone around the
    world.*/

    if ( id == 0 )
    {
        dest = 1;

        source = p - 1;


        tave = 0.0;

        tmin = 1.0E+30;

        tmax = 0.0;

```

```

for ( test = 1; test <= test_num; test++ )
{
/*Just in case, set the entries of X in a way that identifies which iteration of the test is being carried
out.*/

for ( j = 0; j < n; j++ )
{
x[j] = ( double ) ( test + j );
}

wtime = MPI_Wtime ( );
MPI_Send( x, n, MPI_DOUBLE, dest, 0, MPI_COMM_WORLD );
MPI_Recv( x, n, MPI_DOUBLE, source, 0, MPI_COMM_WORLD, &status );
wtime = MPI_Wtime ( ) - wtime;

/* Record the time it took.*/

tave = tave + wtime;

if ( wtime<tmin )
{
tmin = wtime;
}

if ( tmax<wtime )
{
tmax = wtime;
}
}

tave = tave / ( double ) ( test_num );

printf ( " %8d %14.6g %14.6g %14.6g\n", n, tmin, tave, tmax );
}

/* Worker ID must receive first from ID-1, then send to ID+1.*/
else
{
source = id - 1;

```

```

dest = ( ( id + 1 ) % p );

for ( test = 1; test <= test_num; test++ )
{
MPI_Recv( x, n, MPI_DOUBLE, source, 0, MPI_COMM_WORLD, &status );
MPI_Send( x, n, MPI_DOUBLE, dest, 0, MPI_COMM_WORLD );
}
}

free ( x );
}

return;
}

```

## **OUTPUT:**

```

RING_MPI:
C/MPI version
Measure time required to transmit data around
a ring of processes

The number of processes is 3

Timings based on 10 experiments
N double precision values were sent
in a ring transmission starting and ending at process 0
and using a total of 3 processes.

```

N	T min	T ave	T max
100	5.6e-06	4.89e-05	0.0004306
1000	1.97e-05	6.822e-05	0.0002172
10000	6.82e-05	0.00016366	0.000692
100000	0.000534	0.0010807	0.0041201
1000000	0.0077338	0.0110522	0.0391087

```

RING_MPI:

```