MPI Programs

P2p.c

2. Write a MPI program to send the message from a process whose rank=3 to all other remaining processes.

#include <stdio.h>

#include <mpi.h>

#include <string.h>

#define BUFFER\_SIZE 32

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

{

int MyRank,Numprocs, Destination, iproc;

int tag = 0;

int Root = 0, temp = 1;

char Message[BUFFER\_SIZE];

MPI\_Init(&argc,&argv);

MPI\_Status status;

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&MyRank);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&Numprocs);

/\* print host name, and send message from process with rank 0 to all other processes \*/

if(MyRank == 3) {

system("hostname");

strcpy(Message, "Hello India");

for (temp=1; temp<Numprocs;temp++)

{

MPI\_Send(Message, BUFFER\_SIZE, MPI\_CHAR, temp, tag,MPI\_COMM\_WORLD);

}

}

else {

system("hostname");

MPI\_Recv(Message, BUFFER\_SIZE, MPI\_CHAR, Root, tag,MPI\_COMM\_WORLD, &status);

printf("\n%s in process with rank %d from Process with rank %d\n", Message,MyRank,Root);

}

MPI\_Finalize();

}

P2p sum.c

4/2. Write a MPI program to find sum of 'n' integers on 'p' processors using point-to-point communication libraries call

#include <stdio.h>

#include "mpi.h"

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

{

int iproc;

int MyRank, Numprocs, Root = 0;

int value, sum = 0;

int Source, Source\_tag;

int Destination, Destination\_tag;

MPI\_Status status;

MPI\_Init(&argc,&argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&Numprocs);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&MyRank);

if(MyRank == Root){

for(iproc = 1 ; iproc < Numprocs ; iproc++){

Source = iproc;

Source\_tag = 0;

MPI\_Recv(&value, 1, MPI\_INT, Source, Source\_tag,

MPI\_COMM\_WORLD, &status);

sum = sum + value;

}

printf("MyRank = %d, SUM = %d\n", MyRank, sum);

}

else{

Destination = 0;

Destination\_tag = 0;

MPI\_Send(&MyRank, 1, MPI\_INT, Destination, Destination\_tag,

MPI\_COMM\_WORLD);

}

MPI\_Finalize();

}

Broadcast

5. Write an MPI program where the master processor broadcasts a message “HELLO MSRIT” to the remaining processors using broadcast system call.

#include <stdio.h>

#include "mpi.h"

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

{

int rank, i;

MPI\_Init (&argc, &argv);

MPI\_Comm\_rank (MPI\_COMM\_WORLD, &rank);

if (rank == 0) i = 27;

MPI\_Bcast ((void \*)&i, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

printf ("[%d] i = %d\n", rank, i);

// Wait for every process to reach this code

MPI\_Barrier (MPI\_COMM\_WORLD);

MPI\_Finalize();

return 0;

}

Gather.c

3. Write a MPI program where each processor sends an integer number and its rank to the master processor, where the master gathers all the information and prints the data accordingly

#include <stdio.h>

#include <mpi.h>

void main(int argc, char \*argv[])

{

int rank,size;

double param[6],mine;

int sndcnt,rcvcnt;

int i;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&size);

sndcnt=1;

mine=23.0+rank;

if(rank==3) rcvcnt=1;

MPI\_Gather(&mine,sndcnt,MPI\_DOUBLE,param,rcvcnt,MPI\_DOUBLE,3,MPI\_COMM\_WORLD);

if(rank==3)

for(i=0;i<size;++i)

//printf("PE:%d param[%d] is %f \n",rank,i,param[i]]);

printf(" %d %d \n",rank,i);

MPI\_Finalize();

}

Pie collective.c

1. Write a MPI program to calculate and print the value of PI.

#include <stdio.h>

#include <math.h>

#include "mpi.h"

double func(double x)

{

return (4.0 / (1.0 + x\*x));

}

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

{

int NoInterval, interval;

int MyRank, Numprocs, Root = 0;

double mypi, pi, h, sum, x;

double PI25DT = 3.141592653589793238462643;

/\*....MPI initialisation....\*/

MPI\_Init(&argc,&argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&Numprocs);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&MyRank);

if(MyRank == Root){

printf("\nEnter the number of intervals : ");

scanf("%d",&NoInterval);

}

/\*....Broadcast the number of subintervals to each processor....\*/

MPI\_Bcast(&NoInterval, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

if(NoInterval <= 0){

if(MyRank == Root)

printf("Invalid Value for Number of Intervals .....\n");

MPI\_Finalize();

exit(-1);

}

h = 1.0 / (double)NoInterval;

sum = 0.0;

for(interval = MyRank + 1; interval <= NoInterval; interval += Numprocs){

x = h \* ((double)interval - 0.5);

sum += func(x);

}

mypi = h \* sum;

/\*....Collect the areas calculated in P0....\*/

MPI\_Reduce(&mypi, &pi, 1, MPI\_DOUBLE, MPI\_SUM, Root, MPI\_COMM\_WORLD);

if(MyRank == Root){

printf("pi is approximately %.16f, Error is %.16f\n",

pi, fabs(pi - PI25DT));

}

MPI\_Finalize();

}