BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJASTHAN)

CS F422 – Parallel Computing

Lab#3

Note: Please use programs under *Code_lab3* directory supplied with this sheet. Do not copy from this sheet.

The lab has the following objectives:
Giving practice programs for MPI.

Point to point Communication Routine

```
1. #include<mpi.h>
2. #include<stdio.h>
3.
4. double func(double x) {
5.   return (double)x * x;
6. }
7.
8. double Trap(double a, double b, int n, double h) {
9.   double area = (func(a) + func(b)) / 2.0;
10.        for (int i = 1; i <= n - 1; ++i) {
11.          double x = a + i * h;
12.          area += func(x);
13.        }
14.        area *= h;
15.        return area;
16.      }
17.
18. int main() {
19.   int my_rank, comm_sz, n = 1024, local_n;
20.   double a = 0.0, b = 3.0, h, local_a, local_b;
21.   double local_int, total_int;
22.   int source;
23.   MPI_Init(NULL, NULL);
24.   MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
25.   MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
26.   h = (b - a) / n; /* h is the same for all processes */
27.   local_n = n / comm_sz; /* So is the number of trapezoids */
28.   local_a = a + my_rank * local_n * h;
29.   local_b = local_a + local_n * h;
30.   local_int = Trap(local_a, local_b, local_n, h);
31.   if (my_rank != 0) {
32.
33.     MPI_Send(&local_int, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
34.
35.   }
36.   else {
```

```
37.
38.     total_int = local_int;
39.     for (source = 1; source < comm_sz; source++) {
40.       MPI_Recv(&local_int, 1, MPI_DOUBLE, source, 0, MPI_COMM_WORLD,
41.
42.         MPI_STATUS_IGNORE);
43.
44.       total_int += local_int;
45.
46.     }
47.   }
48.   if (my_rank == 0) {
49.
50.     printf("With n = %d trapezoids, our estimate\n", n);
51.     printf("of the integral from %f to %f = %.15e\n",
52.       a, b, total_int);
53.
54.   }
55.   MPI_Finalize();
56.   return 0;
57. }
```

Q?

1. This is a program to calculate the area of an integral using trapezoid method. Compile the program with `` `mpicc point2point.c` ``. How is the program deciding the number of parallel processes?
2. Find out if it's possible to change the number of available parallel processes for the program.
3. Verify mathematically if the area calculated is correct.
4. Try to change the function ("func") and see if it works for different functions.

Collective Communication Routine

```
1. #include<mpi.h>
2. #include<stdio.h>
3.
4. #define SIZE 4
5.
6. int main(int argc, char* argv[]) {
7.   int numtasks, rank, sendcount, recvcount, source;
8.   float sendbuf[SIZE][SIZE] = {
9.     {1.0, 2.0, 3.0, 4.0},
10.          {5.0, 6.0, 7.0, 8.0},
11.          {9.0, 10.0, 11.0, 12.0},
12.          {13.0, 14.0, 15.0, 16.0}
13.        };
14.        float recvbuf[SIZE];
15.        MPI_Init(&argc, &argv);
16.        MPI_Comm_rank(MPI_COMM_WORLD, &rank);
17.        MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
```

```
18.        if (numtasks == SIZE) {
19.          // define source task and elements to send/receive, then perform
   collective scatter
20.          source = 1;
21.          sendcount = SIZE;
22.          recvcount = SIZE;
23.          MPI_Scatter(sendbuf, sendcount, MPI_FLOAT, recvbuf, recvcount,
   MPI_FLOAT, source, MPI_COMM_WORLD);
24.          printf("rank= %d Results: %f %f %f %f\n", rank, recvbuf[0],
   recvbuf[1], recvbuf[2], recvbuf[3]);
25.        }
26.        else
27.          printf("Must specify %d processors. Terminating.\n", SIZE);
28.        MPI_Finalize();
29.      }
```

Q?

1. This is a program to distribute rows of an array to separate processes. Run the program with `` `mpirun -np 4 ./a.out` ``.
2. Why was Scatter used here instead of Bcast?

Derived Data type

```
1. #include "mpi.h"
2. #include <stdio.h>
3. #define NELEM 25
4.
5. int main(int argc, char* argv[]) {
6.   int numtasks, rank, source = 0, tag = 1, i;
7.
8.   typedef struct {
9.     float x, y, z;
10.    float velocity;
11.    int  n, type;
12.  } Particle;
13.  Particle     p[NELEM], particles[NELEM];
14.  MPI_Datatype particletype, oldtypes[2];
15.  int          blockcounts[2];
16.
17.  MPI_Aint    offsets[2], extent;
18.
19.  MPI_Status stat;
20.
21.  MPI_Init(&argc, &argv);
22.  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
23.  MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
24.
25.  offsets[0] = 0;
26.  oldtypes[0] = MPI_FLOAT;
27.  blockcounts[0] = 4;
```

```
28.
29.   MPI_Type_extent(MPI_FLOAT, &extent);
30.   offsets[1] = 4 * extent;
31.   oldtypes[1] = MPI_INT;
32.   blockcounts[1] = 2;
33.
34.   MPI_Type_struct(2, blockcounts, offsets, oldtypes, &particletype);
35.   MPI_Type_commit(&particletype);
36.
37.   if (rank == 0) {
38.     for (i = 0; i < NELEM; i++) {
39.       particles[i].x = i * 1.0;
40.       particles[i].y = i * -1.0;
41.       particles[i].z = i * 1.0;
42.       particles[i].velocity = 0.25;
43.       particles[i].n = i;
44.       particles[i].type = i % 2;
45.     }
46.     for (i = 0; i < numtasks; i++)
47.       MPI_Send(particles, NELEM, particletype, i, tag, MPI_COMM_WORLD);
48.   }
49.
50.   // all tasks receive particletype data
51.   MPI_Recv(p, NELEM, particletype, source, tag, MPI_COMM_WORLD, &stat);
52.
53.   printf("rank= %d   %3.2f %3.2f %3.2f %3.2f %d %d\n", rank, p[3].x,
54.     p[3].y, p[3].z, p[3].velocity, p[3].n, p[3].type);
55.
56.   // free datatype when done using it
57.   MPI_Type_free(&particletype);
58.   MPI_Finalize();
59. }
```

Q?

1. This is a program which uses P2P comms to send a custom data structure (Particle) across MPI processes (each of which have different values). Run the above program with `` `mpirun -np 4 ./a.out` ``.

Process Topology

```
1. #include<mpi.h>
2. #include<stdio.h>
3.
4. int main(int argc, char* argv[]) {
5.   int rank, size;
6.   MPI_Comm comm;
7.   int dim[2], period[2], reorder;
8.   int coord[2], id;
9.
10.  MPI_Init(&argc, &argv);
11.  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
12.  MPI_Comm_size(MPI_COMM_WORLD, &size);
13.
14.  dim[0] = 4; dim[1] = 3;
```

```
15.  period[0] = 1; period[1] = 0;
16.  reorder = 1;
17.  MPI_Cart_create(MPI_COMM_WORLD, 2, dim, period, reorder, &comm);
18.  if (rank == 5) {
19.    MPI_Cart_coords(comm, rank, 2, coord);
20.    printf("Rank %d coordinates are %d %d\n", rank, coord[0],
   coord[1]);fflush(stdout);
21.  }
22.  if (rank == 0) {
23.    coord[0] = 3; coord[1] = 1;
24.    MPI_Cart_rank(comm, coord, &id);
25.    printf("The processor at position (%d, %d) has rank %d\n", coord[0],
   coord[1], id);fflush(stdout);
26.  }
27.  MPI_Finalize();
28.  return 0;
29.}
```

Q?

1. This is a program which generates a virtual topology in the form of a 3x4 array with wrap around (a 2D torus). Run the above program with `` `mpirun -np 12 ./a.out` ``.
2. Can you change the dimensions of the topology?

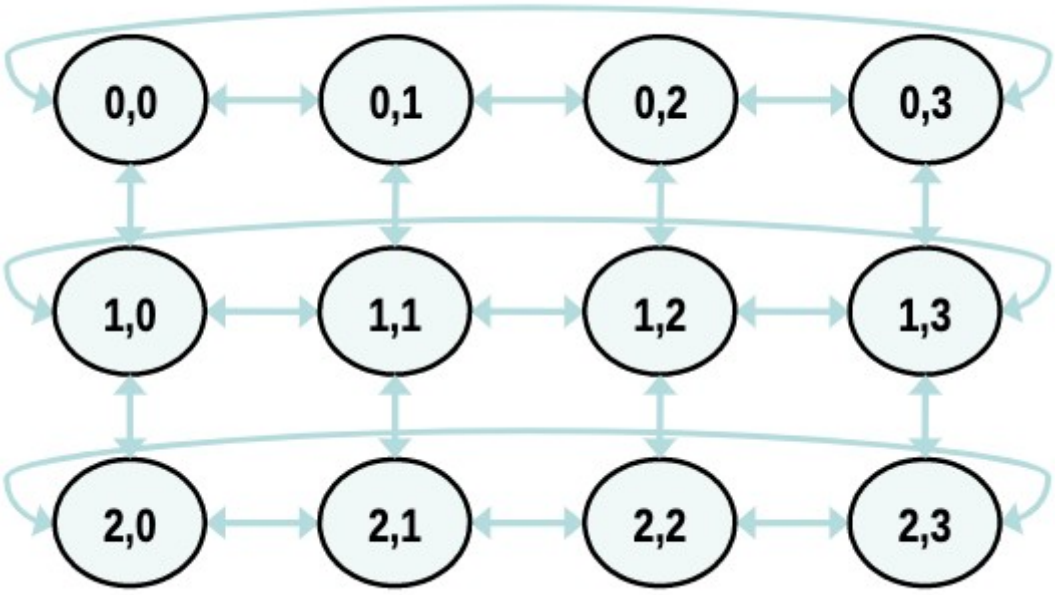

Figure 1: Source: codingame.com

End of lab3