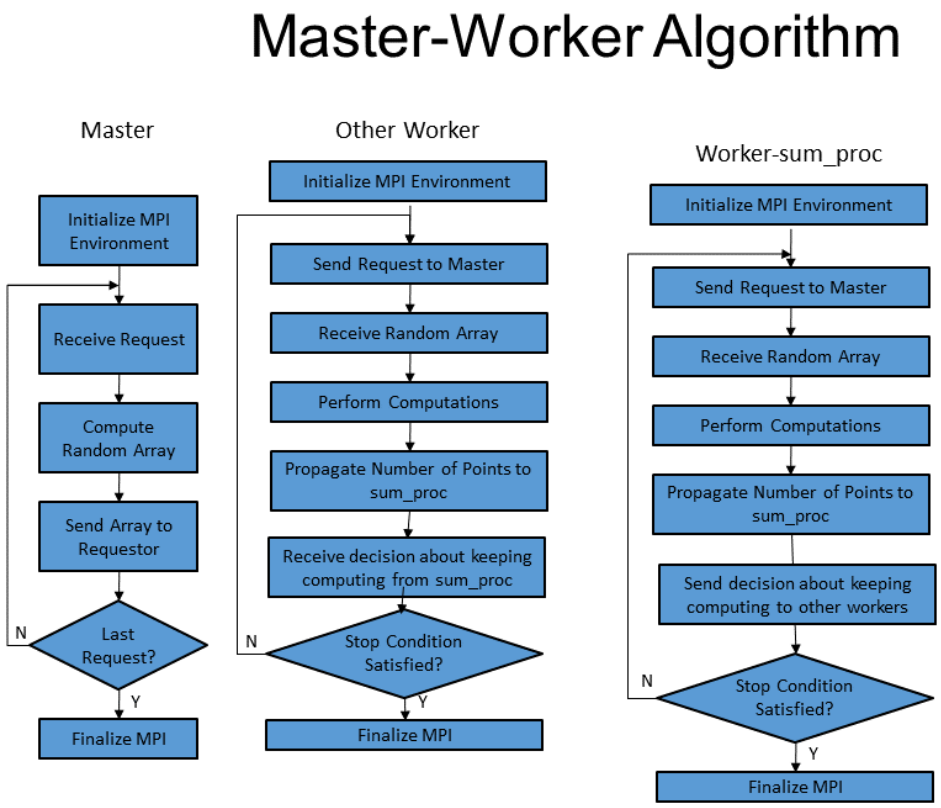
Project 3, due on 04/04.

Project for undergraduate students: Parallel Monte Carlo for π

The base code is ~ zxu2/Public/Proj3_undergrad/ proj3_undergrad.cpp

Method consisting of following steps could be used to approximate π.

1. Inscribe a disk of radius r in a square of side length $2r$. Randomly generate points in the square.
2. Determine the number of points in the square that are also in the disk. Let s be ratio of the number of points in the disk and the number of points in the square.
3. The area of the square is $4r^2$, and the area of the disk is πr^2. So $s = \frac{\pi r^2}{4r^2}$. Therefore, $\pi = 4s$.



1. Complete all missing MPI functions.

2. Use $epsilon = 0.001, 0.0001\ and\ 0.00001$ to do the calculation respectively. For each computation, use 4 and 8 processors respectively. Modify the script "paral_HPCC.sh" to submit your runs. The script is under directory: zxu2/Public/ACMS40212/MPI_basics. Find the overall the wall clock times spent by the program respectively. Make a table to list the results.

Hand-In. Turn in the hardcopy of all your source code, and the report which contains results and a description of your implementation on point-to-point communication. Email the source code. Use the following title for your email: acms40212S16-Proj3-your-ND-ID.

Project for graduate students: Parallel subdomain decomposition

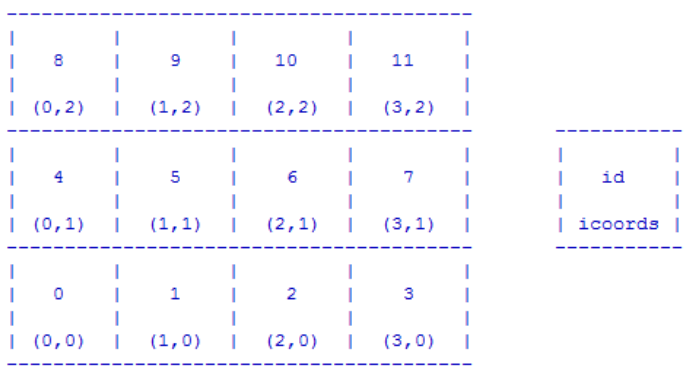
The base code is at ~ zxu2/Public/Proj3_grad/

The code consists of following files: **comput_2d.h int.h, comput_2d.cpp mesh.cpp proj3_main.cpp.**

The mesh data files are: ~ zxu2/Public/Proj3_grad/Mesh/rect_mesh.*

1. Defines subdomains.

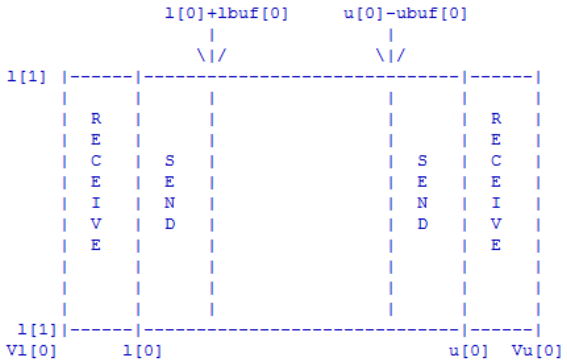
RECT_GRID is used to save this information. This information is initialized in constructor of class Compute_2d.



2. The workload assigned to each process is estimated in function Compute_2d::estimate_workload(). The sub-region $[x_{i,l}, x_{i,u}]$ on ith process is saved in L[0] and U[0] of variable rect_grid of RECT_GRID type.

3. Each process reads the mesh data. Use a linked list to save triangles defining the mesh(Use code of project 2 for this purpose). Then according to the subdomain size defined by (VL[0], VU[0]) by (VL[1], VU[1]), delete triangles outside this region.

4. Implement parallel_communication () function.



Let [l[0],u[0]] be a subdomain assigned to a process. For convenience of computation, a virtual domain is defined to hold the ghost points for updating solutions defined on grid points within [l[0],u[0]]. This virtual domain is defined as [vl[0],vu[0]]. Here vl[0]=l[0]-N*Δx, and vu[0]=u[0]+N*Δx. N is the number of ghost points.

Inside pp_send_interior_fields() function, find triangles inside the region marked by "SEND" in the above figure, send ids of these triangles to the adjacent subdomain whose buffer zone is marked by "RECEIVE" and overlaps with this marked by "SEND" region.

Inside pp_receive_interior_fields() function, receive ids of triangles sent by the pp_send_interior_fields() function.

5. Use 4 and 8 processors respectively to test the code. Modify the script "paral_HPCC.sh" to submit your runs. The script is under directory: zxu2/Public/ACMS40212/MPI_basics. Find the overall the wall clock times spent by the communication part of the program and the part of the program for creating list of triangles inside the subdomain, respectively. Make a table to list the results.

```
void parallel_communication (double *soln)
{
    int          myid, side;
    int          me[3];

    MPI_Comm_rank(MPI_COMM,&myid);

    for (side = 0; side < 2; ++side)
    {
        MPI_Barrier(MPI_COMM);
        pp_send_interior_fields(myid, side,soln);
        pp_receive_interior_fields(myid ,(side+1)%2,soln);
    }
}

void   pp_send_interior_fields(
      int          *me,
      int           side,
      double       *soln)
{
      int           myid,  dst_id, ntasks;

     MPI_Comm_rank(MPI_COMM,&myid);
     MPI_Comm_size(MPI_COMM_WORLD, & ntasks);
     dst_id = (myid + 2*side - 1);
     if(dst_id < 0)
         dst_id = ntasks-1;
     if(dst_id>= ntasks)
         dst_id = 0;
     /* Next collect soln points to be sent and call MPI_bsend() to send the data
        to the process with rank dst_id */
}

void   pp_receive_interior_fields(
      int          *me,
      int          side,
     double       *soln)
{
      int           myid,  src_id,  ntasks;
       MPI_Comm_size(MPI_COMM_WORLD, & ntasks);
      MPI_Comm_rank(MPI_COMM,&myid);
      src_id = (myid + 2*side - 1);
      if(src_id < 0)
         src_id = ntasks-1;
     if(src_id>= ntasks)
         src_id = 0;

     /* Next call MPI_Recv() to receive the data
        from the process with rank src_id */
}
```

Hand-In. Turn in the hardcopy of all your source code, and the report which contains results and algorithmic notes on both computation and communication. Email the source code. Use the following title for your email: acms40212S16-Proj3-your-ND-ID.