CS 6230 – High-Performance Computing and Parallelization
Project 1: Due February 24, 2011

The purpose of this homework is to expose you to parallelizing a serial algorithm. You will be required to deduce an appropriate parallelization strategy, model that strategy mathematically to predict anticipated speedup, to implement your strategy in MPI, and to validate your model by running experiments to see if your solution does indeed scale as you have predicted (and if not, to discuss why).

The starting point for this project is the following paper that you can obtain from the instructor's website:

www.cs.utah.edu/~kirby/publications.html Paper Number 29:

Michael Steffen, Robert M. Kirby and Martin Berzins, "Analysis and Reduction of Quadrature Errors in the Material Point Method (MPM)", *International Journal for Numerical Methods in Engineering*, Vol. 76, Issue 6, pages 922-948, 2008.

Building blocks of a serial implementation written by the TA will be provided both in algorithmic form (in this document) and in code (available on the course website).

Your task is to parallelize this code using MPI using two different strategies:

1. Spatial decomposition of the grid so that each processor is assigned some patch of cells for which it is responsible. Communication must be used for resolving both grid and particle dependencies between patches.
2. A worker-pool approach in which processes update cell information that is farmed off to them but a single master process is responsible for assembling all information and making the final computations necessary to accomplish a time step.

Several important things to note:

- A simulation consists of many time steps. You may need to change the total number of time steps run in order to see asymptotic trends from your model (that is, you want to run for sufficient time that startup time appears negligible).
- Your parallel solution should always give the same answer as the serial solver when run with identical starting conditions, parameters, etc.

Report

1. Introduction: Describe the problem you are trying to solve, appropriate background information, and the goals of the assignment.

2. Mathematical Description of the Algorithm: A brief description of the mathematical building blocks of the problem.

3. Algorithmic Description of the Serial Algorithm. Describe the serial algorithm.

4. Algorithmic Description of the Parallelization of the Algorithm. Describe your parallelization strategy (or strategies).

5. Mathematical Model to Describe Anticipated Performance. Describe a mathematical model of the anticipated performance (speedup) of the serial algorithm.

6. Results. Accomplish simulations for all strategies for 2, 4, 8, 16, and 32 processors. Provide results for both strong and weak scaling.

7. Discussion and Conclusions. Discuss your results. Did your mathematical model adequately describe the behavior you encountered when running your simulations?

Description:

The Material Point Method is used to compute the deformations of the solid bodies when acted upon by external forces. In order to compute this, we visualize the body as discrete points which together form the body. The number of material points depends on the number of divisions of the body's volume. Then, at each time step, we find the various parameters of the material points such as the velocity, position etc and hence track the particle's motion. In order to perform this, we consider a mesh on top of which the body is considered to move in a 2D space.

In order to algorithmically write the steps involved in this process and hence compute the output we start by defining the following structures in C.

```
struct material {
  double x,y;   //position

  double ux,uy; //displacement

  double vx, vy; //velocity

  double grad_velocity[2][2]; //gradient of velocity

  double ax, ay; //acceleration

  double mass;

  double vol;   //volume

  double basis_fn[4]; //Array containing the basis_fn value for  the particle for 4
                      //nodes surrounding it


  double grad_x_basis_fn[4]; //This contains the gradient values of the basis
                             //function present above with respect to "x"

  double grad_y_basis_fn[4]; //This contains the gradient values of the basis
                             //functions present above with respect to "y"

  double stress[2][2]; //The stress of the material

  double change_Fp[2][2]; //Change in deformation gradient

  double Fp[2][2]; //Deformation Gradient
```

```
  int cell_i,cell_j; //i and j value of the containing cell

  double E; //Young's modulus

  double body_force_x , body_force_y;
};
```

In the above structure, properties of the material point are defined.

```
struct IJ
{
  int x,y; //These are  the location of the points in the 2D array of the  "struct
           //material" instance created.
};


/*A mesh in a 2D space consists of collection of cells.
A cell can be considered a square with 4 nodes {0,1,2 and 3}
In the structure below, we denote a cell by the node 0.
Since we have a constant spacing "h" we can derive the rest of the nodes.
We use an instance of this "struct mesh" both in the context of a cell and in the
context of the "node 0" of the cell.
*/

struct mesh {

  double x,y; //x and y position of the node 0 of a cell

  struct  IJ  points_list[1000]; //List containing the information about the points
                                 //contained in the cell

  int num_points; //Number of points contained in a cell

  double f_int_x,f_int_y; //Internal forces

  double f_ext_x,f_ext_y; //External forces

  double a_x , a_y; //Acceleration

  double v_x, v_y; //Velocity

  double node_mass; //Mass of a node
};
```

struct IJ is a book-keeping structure whose instance is declared in *struct mesh*. It holds the (x,y) coordinates of the points present in the points_list of *struct mesh*.

Particle and Mesh Spacing:

We represent the particle spacing in the material by "b" and the spacing between nodes in a mesh by "h".

Steps:

Once this is setup, we can look into the steps involved in this method. In this method, the computations proceed in steps of time (ie) at time t_1 we calculate the position, velocity etc for the particle and then proceed to time t_2 and calculate the same based on the values in the previous time step.

1. We find which material point is present in which grid cell. Once we associate a grid cell with a point by updating the *points_list array* of *struct mesh* and increment the counter *num_points.*

2. For each material point in the domain, we calculate the basis_function of that point with respect to each of the four nodes of the cell containing the material point. The result is stored in "*double basis_fn[4] ".* This information will be used, by the nodes when we calculate the velocity and other quantities of the nodes. The 1-dimensional example of the basis function can be found in **eqn(14)** of reference.

3. Now let us calculate the mass of each node, given by "*double node_mass"* in struct mesh. The formula to calculate this is given in a generalized form in **eqn(1)** of the reference paper.

4. For each node we calculate the velocity, given as "*double v_x, v_y;"* in struct mesh. The formula to calculate this is given in **eqn(2)** in the reference.

5. For each particle, similar to the basis_fn calculation, also calculate the gradient of the basis function, by substituting the value of x and y respectively after differentiation of "*basis_fn_x and basis_fn_y"*. A one-dimensional equivalent is found in **eqn(15)** of the reference.

6. Velocity gradient, given by *"double grad_velocity[2][2]"* which is a (2X2) matrix is calculated using eqn **(3).**

7. The deformation gradient given by "*double Fp[2][2]*" is calculated after calculating the change in deformation gradient given by "*double change_Fp[2][2]*" as in eqn **(10) and (11)**

8. Now calculate the internal force at each node given by "*f_int_x and f_int_y*" and and external force at each node given by "*f_ext_x and f_ext_y*". This is given by eqns, **(4)** and **(13)** of the reference.

9. Calculate the acceleration at each node a_x and a_y, given by eqn **(5)** of reference.

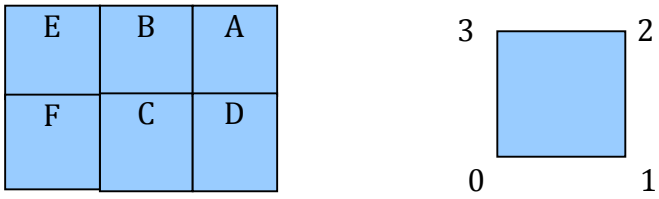
10. Now calculate the velocity of the particle vx and vy using the velocity of the nodes v_x and v_y along with the corresponding gradient value of the basis function of the node for that particle. This is given by eqn**(8)** of the reference.

11. Now the change in position of the particle for x and y coordinates is calculated as in eqn **(9)** of the reference.

12. The above steps (1-10) are repeated for a given number of iterations.

Information on calculations at nodes and at particle position:

In the above algorithm, whenever we calculate the mass and velocity of the node, we iterate among the particles present in the cells as explained with the following figure.



The above figure depicts four cells (A, B, C and D). In the above figure, we have nodes (0,1,2 and 3) present for each cell, as given on the right side figure.

When we calculate the mass and velocity of node 0 of cell A, we iterate through all the particles present in the cells A, B, C and D. Similarly when we calculate the velocity and mass at node 0 of cell B, we iterate through particles present in cells B, E, F and C.

When we calculate the velocity and other values for the particles, we iterate through the 4 nodes of the cell containing the particle and consider the required mass, velocity values etc of those nodes.