Homework #6 (Due Nov. 14)

Skeleton code for this problem can be found in directory hw6 in our git repository. You can use either "git pull" or "git clone" as usual.

You need to modify particles.c to contain your implementation of the parallel algorithm described below. I've provided you with a serial implementation and skeleton code for the parallel implementation. You are required to write a report (several pages) comparing this algorithm with various sized problems and cluster sizes. Write this as if you are submitting it as a workshop paper.

Since this homework requires large-scale data, you need to start the assignment early. It is crucial to finish the coding and debugging days before the deadline, so you allow enough time for the runs to complete.

To gather the experimental results, in particular the large-scale data, modify and use the provided example batch scripts:

```
$ qsub submit_parallel.batch
```

This homework will be worth double the points of previous homework, it will be graded roughly 50% code correctness and 50% report.

Particle Interaction

A very common problem in scientific simulations is to solve a n-body problem. Such problem involves a system of particles interacting with each other via *van der Waals* forces, using a simplified model, in a 2D world. The calculation for the force f between particles (x_1, y_1) and (x_2, y_2) is given by the following formulas:

$$f = \frac{A}{r^6} + \frac{B}{r^{12}} \qquad r_x = x_1 - x_2 \qquad r_y = y_1 - y_2$$

$$r = \sqrt{r_x^2 + r_y^2} \qquad f_x = \frac{f * r_x}{r} \qquad f_y = \frac{f * r_y}{r}$$

where A and B are constants.

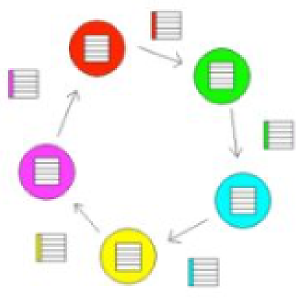
There will be an odd number of processors. Each processor will own a subset of the total particles. Each particle has a (x, y) position, mass, and force associated with it. The MPI program calculates the force on each particle. The processors will be arranged in a ring such that it will only receive messages from the previous processor and send to the next processor (with the exception of the last step). The particles are initialized with random locations and forces unless those values are contained in a provided text file.

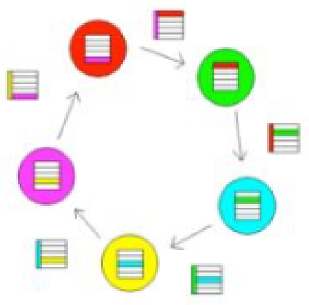
If an input file is provided, rank 0 will be the only one reading the particle information. You must use collective communication operations to distribute the particles among the processors. Similarly, after the simulation is finished, rank 0 will collect all particles and output their information. Again, collective communication operations must be used to collect particles.

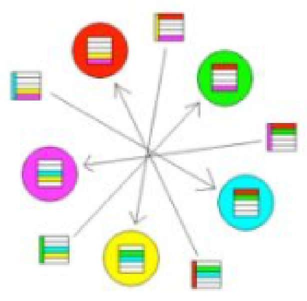
The simulation algorithm is composed of the following steps:

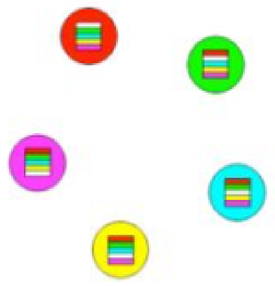
1. Each processor sends its particles to the next processor (location, mass, and running total of forces so far).
2. Upon receiving a set of particles, a processor calculates the resulting forces from the local particles to the remote particles. You must call function compute_interaction with locals and remotes (in that order).
3. Each processor updates total force on local particles and remote particles.
4. Each processor sends the remote particles, along with the calculated forces, to the next processor.
5. Repeat previous steps for (p - 1)/2 stages.
6. Each processor sends remote particles back to their original processor.
7. Each processor calculates the forces resulting from its own particles. You must call functions merge and then compute_self_interaction (in that order).
8. Each processor finishes execution.

Figure below presents an overview of the algorithm. The system contains 5 processors in this case and each processor is represented by a circle with a distinctive color. Messages carrying particles are rectangles with a vertical line representing the particles being transferred in a message. Horizontal lines in a message represent the effect of other particles whose effect has been calculated. For instance, the message in (b) going from the red to the green processor is carrying the purple particles and has collected the effect of interacting with the red particles.


(a) Step 1.


(b) Steps 2-4.


(c) Step 6.


(d) Steps 7-8.

The format of messages requires the following information for each particle to be passed:

1. Location: x and y.
2. Running sum of forces in x and y
3. Mass

The program should take at least one command line parameter for the number of particles:
\$ mpirun -np *p* particles *N* [filename]
The program should divide *N* as evenly as possible among the total number of processes.

Submission:
You should create a report with the following sections, at least 2-3 pages with tables:

1. A general strategy of the parallelization effort. Why did you choose those MPI operations to parallelize the program?
2. A speedup analysis. Using an interesting value of N, the number of particles, report the speedup for a number of cores ranging from {3, 15, 63}. Repeat each experiment at least 3 times and report the average value.
3. An efficiency analysis. Using an interesting value of N, the number of particles, report the efficiency for a number of cores ranging from {3, 15, 63}. Repeat each experiment at least 3 times and report the average value.
4. A description of the performance bottlenecks. What is preventing the program from getting higher speedup?

Place your report in hw6 directory. Then compress your hw6 directory in a single tarball (.tar) file, and name it as follows: "PittID_hw##.tar". For example, if your pitt ID is pit11, and you are submitting the 8th homework, then, the tar file should be named: "pit11_hw08.tar". Submit your tarball file to CourseWeb.