N-body simulation (2013 FA)

An `n-body simulation` models the movement of objects in space due to the gravitational forces acting between them over time. Given a collection of n bodies possessing a mass, location, and velocity, we compute new positions and velocities for each body based on the gravitational forces acting on each. These vectors are then applied to the bodies for a small period of time and then the process repeats, creating a new vector. Tracking the positions of the bodies over time yields a series of frames which, shown in succession, model the bodies' movements across a plane.

To understand how the acceleration function works, we need to review a few basic facts from physics. Recall that force is equal to mass times acceleration ($F = m \times a$) and the gravitational force between objects with masses m_1 and m_2 separated by distance d is given by $\frac{G \times m_1 \times m_2}{d^2}$ where G is the gravitational constant (given by `Simulation.g`). Putting these two equations together, and solving for a, we have that the magnitude of the acceleration vector (due to gravity) for the object with mass m_2 is $\frac{G \times m_1}{d^2}$. The direction of the acceleration vector is the same as the direction of the unit vector between from the subject body to the gravitational body; in this case, the acceleration direction is from m_2 to m_1. Note that this calculation assumes that the objects do not collide.

Given accelerations for each body, we move the simulation forward one time step, updating the position `p` and velocity `v` of each body to `p + v + a/2` and `v + a` respectively, where `a` is the acceleration of the body.

This algorithm fits nicely into the MapReduce framework. Accelerations for each body can be computed and applied in parallel: map across the bodies to get the accelerations on each due to every other body, then apply each acceleration vector to get a new position and velocity for the body.

Provided code and data

We have provided some sample data to get you started. The `data` directory contains files describing the initial conditions of various simulations. The function `Simulations.read_data` can be used to parse these files. You can output the results of your simulation into a file using the `Simulations.write_transcript` function.

The transcripts produced by `write_transcript` can be viewed using the supplied viewer `bouncy.jar`, by running

```
java -jar bouncy.jar
```