Grant Mitchell

9/28/17

Project Proposal

1. What will the program do/compute? Briefly describe the problem the program addresses.

   My program will simulate states of interstellar bodies under the influence of gravity. The bodies will be stored as memory arrays that contain mass, x_pos, y_pos, x_vel, y_vel, radius. These values will all be calculated over time, with each value changing based on Newtonian gravity. It will be configurable for different masses of bodies, different numbers of bodies, and different lengths of time. The data will all be computed in a c algorithm, and will be visualized with a python script.

2. Where is parallelism expected to improve performance? Briefly describe where and why adding parallelism to the code is expected to improve performance.

   Parallelism will greatly improve the speed of the calculation. Each core will take on the task of calculating the next timestamp for a single interstellar body. With thousands of bodies, having the calculation done across 32 or more cores should significantly increase performance. I also will have the file writes of each timestamp be handled on its own thread. With this setup, computation will be allowed to continue while writing to a file, a computationally intensive activity, can happen on its own.

3. How will you prove performance is better? Briefly describe how you intend measure/analyze the performance of your program.

   I plan to do two things to measure the performance of my algorithm. In terms of actual experimental data, I want to have a serial version of my code, and a parallel version. Then I can measure the difference on small inputs and see if the difference is significant. I also plan to do some theoretical work on the serial parts of my program versus the parallel part. The interstellar body update is the parallel part of the program, and so comparing the number of pre and post processing steps compared to the number of parallel steps for a certain input, should allow me to calculate the speedup and efficiency of my parallel algorithm.