Homework 4
N-body Assembly Line

Geoffrey Ulman
CSI702

March 2010

1 Design

My parallel N-body gravitational potential code uses the assembly line programming paradigm. Each node calculates the gravitational potential for $\frac{n}{m}$ particles where n is the number of particles and m m is the number of nodes. These will be refered to as *host particles.* In adition to the $\frac{n}{m}$ host particles, each node has $\frac{n}{m}$ *guest particles.* The algorithm has n iterations. During each iteration, the node calculates the gravitational potential between its host particles and guest particles then passes its guest particles to one of its neighbors while receiving another set of guest particles from its other neighbor. After each node has had the particles from all other nodes as guest particles, it reports its gravitational potential values back to node 0.

2 Challenges

Text...

3 MPI Commands

Like all MPI programs, the commands `MPI_Init`, `MPI_Comm_size`, `MPI_Comm_rank`, and `MPI_Finalize` were used to initialize and get basic information about the MPI environment. Because data was being passed in a ring, `MPI_Cart_create` was used to create a new Cartesian comm group and `MPI_Cart_shift` was used to determine the coordiates of the nodes neighbors. In addition, `MPI_Cart_coords` was used to determine whether the node was in an even or odd positioin in the new Cartesian comm group. Data was sent using `MPI_Isend` and `MPI_Irecv` so that processing of the last group of host particles could be performed while the new group of host particles was being received. Finally, completion of the communications in each iteration was checked with `MPI_Waitall`.

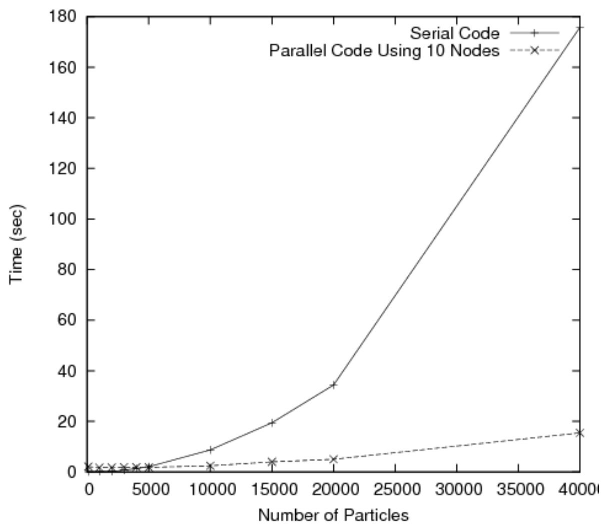

Figure 1: Parallel and Serial Timing Results for Variable Particle Count

4 Performance

Text...

5 Performance Comparison

Text...

6 Output Comparison

The output of the serial and parallel codes are extrememly similar, however in most cases they were not identical. I used the `MPI_FLOAT` data type to transfer data between nodes and c float to store them in the nodes. The error in the final calculation compared with the serial code was .

References

[1] Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language*, Prentice Hall PTR, New Jersey, 2009.

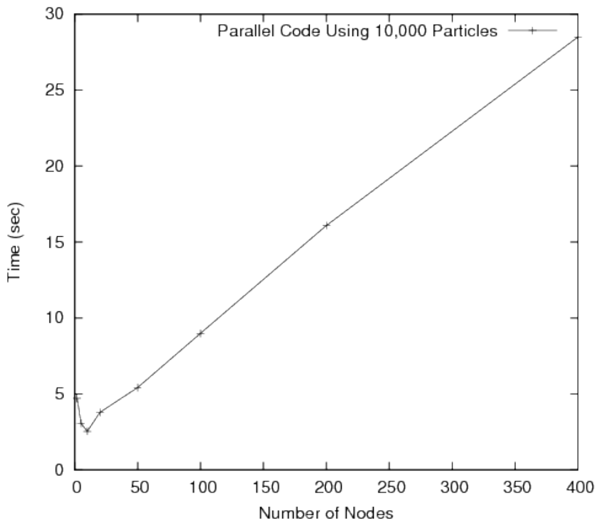

Figure 2: Parallel and Serial Timing Results for 10,000 Particles

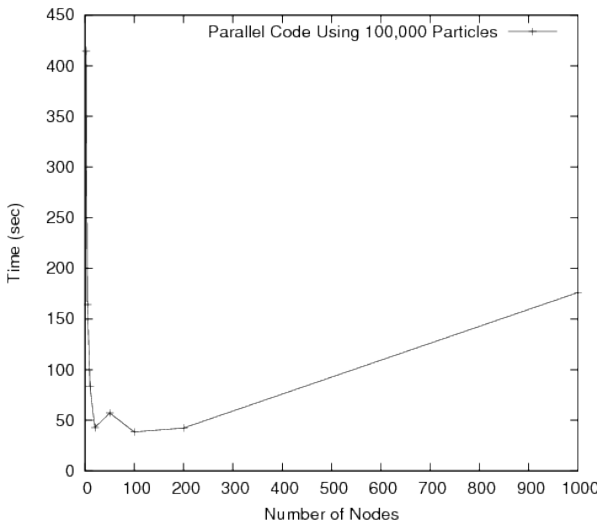

Figure 3: Parallel and Serial Timing Results for 100,000 Particles