CS 380P Parallel Systems — Spring 2013	Programming Assignment #5
Barnes-Hut in MPI	Due March 27, 2013

In this assignment, you will solve an N-body problem using the Barnes-Hut algorithm. You will be forced to deal with an irregular data structure, ie, something other than an array, and you will have to deal with dynamic changing data as the bodies move during the simulation. We encourage you to think carefully about your data structure, because your choice will be central to your solution. You may work with a partner if you wish.

1 Background

The particular N-body problem that you will solve will be an astrophysical simulation that models the continuous interaction of each body with every other body. In this case the interactions are caused by the gravitational force that each body exerts on every other body, where each body could represent a star, a planet, or a galaxy. As your simulation proceeds, the forces will cause the bodies to move, so this is a dynamic computation that requires an irregular data structure.

2 Algorithm

There are several algorithms for solving the N-Body problem. The simplest solution directly computes the n^2 interactions that exist between every pair of bodies. The complexity of this approach is $O(n^2)$, where n is the number of bodies in the system. You will instead use a superior solution, the Barnes-Hut method, which takes $O(n \log n)$ time; this solution computes direct interactions for pairs of bodies that are sufficiently close; for the effects of far away bodies, the algorithm approximates a group of bodies by a single body whose center of mass is the same as the center of mass of the group of bodies. The following papers describe the algorithm in more detail and describe various optimizations.

- A. Y. Grama, V. Kumar and A. Sameh, "Scalable Parallel Formulations of Barnes Hut Method for N-body Simulations," *Proc. of Supercomputing '94*, pp. 439-448, Nov. 1994.
- J. Singh, C. Colt, T. Totsuka, A. Gupta, and J. Hennessy. "Load Balancing and Data Locality in Hierarchical N body Methods," *Journal of Parallel and Distributed Computing*, 1994.
- M. Warren and J. Salmon. "Astrophysical N-body Simulations Using Hierarchical Tree Data Structures," *In Proceedings of Supercomputing Conference*, 1992.
- `http://developer.nvidia.com/content/cudazone/cuda_sdk/Physically-Based_Simulation.html` Nvidia CUDA SDK example – Refer to the N-body simulation.

3 Algorithm Structure

The algorithm consists of two phases. The first phase constructs a tree that represents a spatial partitioning of the bodies in the system. Each interior node in the tree represents the center of mass of the bodies in the subtree rooted at that node. The second phase computes the force computations for each body in the system by using the MAC (multipole acceptance criteria), which determines whether the bodies in a subtree are sufficiently far away that their net force on a given body can be approximated by the center of mass of the bodies in a subtree. The velocity and position of the body is then calculated via a Leapfrog-Verlet integrator. Since we are approximating the effect of group of bodies by their center of mass, this algorithm takes $O(n \log n)$ steps.

The TA will be posting a snippet of C code that performs the actual Leapfrog-Verlet integrator. Alternatively, you can refer to the CUDA example (SDK/examples/nbody), which has code that contains this calculation.

4 Parallel Programming Paradigm

You should use **MPI** for this assignment. For better performance, you can use a hybrid of MPI and Pthreads to better utilize the individual Lonestar nodes.

5 Input/Output

Your program should accept the following four command line parameters, with types indicated in parentheses:

1. number of iterations (double)
2. timestep (double)
3. input filename (char *)
4. output filename (char *)

Both the input and output files will have the following format:

- The number of bodies (n)
- A list of n tuples:
  - x position
  - y position
  - mass
  - x velocity
  - y velocity

  You may assume that all initial velocities are 0 in the input file.

6 Details

As usual, you may develop your code on any platform, but we will use the Lonestar and Stampede clusters at TACC to grade your solutions, so be sure that what you submit works on Lonestar and Stampede.

What to Turn In

This assignment is due at 11:59pm on the due date. Use the **turnin** program to submit your solution, which should include the following:

1. A written report of the assignment in either plain text or PDF format. Please explain your approach, present performance results, discuss any insights gained, etc.
2. Your source code, instructions to build it on Lonestar and Stampede, and instructions to run the programs (along with the arguments, etc).

More detailed instructions will be made available later on our Piazza page.

Acknowledgments. This assignment was prepared by Karthik Murthy.