Introduction

This report outlines and analyses an implementation of a message passing parallel algorithm for k-means++ clustering and the n-body simulation.
The algorithm is implemented with Open MPI where a number of nodes collaborate and passes messages to each other to carry out the task required.
The task is described as follows (n denotes the total number of points and m denotes the total number of nodes):

1. n points are sampled by all m nodes from a Gaussian mixture model.
2. All nodes collaborate to cluster the points into m clusters using k-means++.
3. Each cluster is moved into one single node so that a single node contains all points of a cluster.
4. The nodes collaborate with each other to perform n-body simulation where each point represents a body with a mass of 1 and bodies are attracted to each other via gravity.

The implementation for sampling all n points and clustering them with k-means++ are all rather naive and straightforward. The interesting part is the n-body gravity simulation.

To reduce the simulation time, the Barnes-Hut simulation approximation algorithm is used so that points over a certain distance away are simplified into a single center of mass when calculating the force they exert on another point. To reduce the communication overhead, each node constructs a Barnes-Hut tree and then prune the tree for each other nodes so that the tree only contains the nodes that are actually needed for the calculation of points in each other node. This pruned tree will from now on be referred to as the partial tree.

For node A to construct a partial tree to be sent to node B, we would like to make sure that any node of the tree where $s/d < \theta$ is kept, where s is the width of the region the node of the tree represent, d is the distance between the center of mass of the node and the closest point in cluster B to the center of mass to the node, and θ is a predefined constant. However, we do not want to calculate the closest points in cluster B to every node in the Barnes-Hut tree for cluster A and that would be slow in both computation time and communication overhead. Hence, we instead take the closest point in cluster B with respect to the center of mass of all points in cluster A, and calculate a hyperplane that is perpendicular to the line through the center of mass and the closest point, that goes through the closest point. Assuming that all points in each cluster are closer to their center of mass than any points in other clusters, the distance from any point within cluster A to this hyperplane should be shorter than the distance from point within cluster A to any point within cluster B.

The pseudocode describing the algorithm is shown in Algorithm 1.

While in theory, this algorithm should be faster than naively parallelizing Barnes-Hut algorithm by transferring all points to a root cluster and calculating one single Barnes-Hut tree to be sent to all clusters. In practice, it is possible that all the overhead of calculating m partial trees (again here m is the total number of computing nodes), encoding them into an array of bytes to be sent over to each node and decoding the bytes into a partial tree, plus the extra computing time of creating m^2 partial trees, might have outweighed the extra communication overhead of setting all points to all nodes. It would be interesting to compare the performance of the current implementation against the alternative described above.

Methodology

Experiments

Analysis & Discussion