Rhiannon Malia - rmalia
Shannon Williams - srw

Lab 4: OpenMPI

Outline

1. Our Design

Our design is rather simple. We have an abstract class to encapsulate the necessary distance and mean functions as well as the Sequential and Parallel versions of the cluster algorithms (since these are the same for both set types essentially).
The cluster algorithms take a List of data and integers k and mu, and start computing by choosing random centroids from the data. We use Lists specifically because Collections.shuffle makes it easy to randomly select points from the data. We then use the k-means clustering algorithm to build the similar clusters from these centroids over mu iterations.
The DNA clustering algorithm can create new centroids by considering the strings in the data set and building a new centroid tag by tag by taking the most common letter at that position in the string. For example, for s1 = TAGC, s2 = TGAA, and s3 = GDAC, we get TGAC as the new centroid.
The Data Set clustering algorithm is a little easier and forms new centroids just by finding the average of the points in the data set.
Over the mu iterations, we can refine the centroids and clusters to better match their constituents.
The Sequential version obviously does this process in a sequential fashion across the entire data set and over all iterations, thus taking a lot of time. We created both separate classes to run the DNA and Data clustering algorithms in sequential order to make it as simple as possible. Given more time, we probably could have made a more generic sequential runner; however, both classes were built off the abstract function in ICluster.
The Parallelized version farms out tasks over the data to different processes so that it can be done simultaneously across smaller chunks to save time.
We can generate data to run these versions on by using our own data set generators for DNA and Data Points, which generate using the Random class, Lists of arrays of data of the specified number and length which are saved on the network at a given filename, which makes it easy to distribute and read the generated data anywhere over AFS.

2. Our Classes

1. **ICluster**
   - an abstract class for k-means clustering
   - contains mean and distance functions
   - contains Sequential and Parallelized Cluster functions
2. **DNACluster**
   - implementation of DNA String version of iCluster

    - contains a generating function to create Lists of randomized DNA strands of specified **length** and **number** to be saved at a given **filename**
3. **DataCluster**
    - implementation of Data Points version of iCluster
    - contains a generating function to create a List of data points (int arrays) of a specified **length** and **number** within a given **range** of numbers to be saved at a given **filename**
4. **GenDNACluster**
    - Generator for DNACluster, which utilizes the DNACluster's generate function
5. **GenDataCluster**
    - Generator for DataCluster, which utilizes the DataCluster's generate function
6. **ParDataCluster**
    - Runs the parallelized version of DataCluster using OpenMPI
7. **ParDNACluster**
    - Runs the parallelized version of DNACluster using OpenMPI
8. **SerDataCluster**
    - Runs a sequential version of DataCluster on the local machine
9. **SerDNACluster**
    - Runs a sequential version of DNACluster on the local machine

3. Implemented Features and Bugs

Implemented Features:

1. Abstract class describing the basic functions for sequential and parallel computing as well as mean and distance functions
2. Parallel and sequential versions of the two k-means clustering
3. DNA and DataPoint data set generators (contained within the DNACluster and DataCluster)
4. Testing classes for generating and running parallel and sequential versions of DNA and Data Point clustering

Warnings:

1. Failure to use OpenMPI enabled ghc machines results in the project not compiling. This isn't a fault of the code, but is meant as an explanation.
2. If for some reason it converges to fewer centroids than requested it will only print out those it found.

4. How to Build and Run our Project

Building Our Project

We have provided a makefile to compile the associated files. Calling 'make' within the project folder will compile all the code for proper use and testing.

You can run the OpenMPI program in the following way:
First login,locally or via ssh,to one of the machines. From there, ssh to several other machines and ensure that your ~/.ssh/known_hosts file is up-to-date. For OpenMPI to work, you need to be able to *ssh* to the machines you will be using without entering a password.

You can generate data by running for either DNA or Data Points:
java GenDNACluster <len> <num> <filename>
java GenDataCluster <len> <num> <filename>
where **len** is the length of each record, **num** is the number of records you want and **filename** is where you want to

save the resultant data.

You will need to be able to do this with the machines provided in the **hosts.txt** to run the MPI processes on without having to log in with your password. While logged onto an Andrew GHC machine, we suggest ghc02, call the following command in the terminal : **mpi -np <num> -machinefile hosts.txt java -classpath ParDataCluster <filename> <k>,** where **num** is the number of processors you want on each host to run, **filename** is the path to the data saved on AFS, and **k** is the number of desired clusters. This runs the parallelized version of Data Point Clustering.

You can run the parallelized version of DNA Clustering analogously, by calling **mpi -np <num> -machinefile hosts.txt java -classpath ParDNACluster <filename> <k>,** where **num** is the number of processors you want on each host to run, **filename** is the path to the data saved on AFS, and **k** is the number of desired clusters. This runs the parallelized version of DNA clustering.

You can run the sequential versions of both DNA and Data Point Clustering by calling:
java SerDNACluster <filename> <k>
java SerDataCluster <filename> <k>
where we pass it the **filename** where the data is stored and a **k,** number of clusters we want to make.

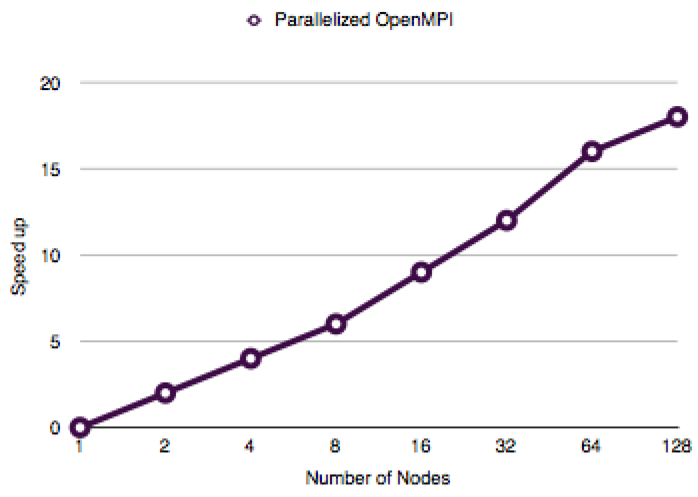
Both versions of DNA and Data Clustering print out the centroid values to show where they centered their clusters.

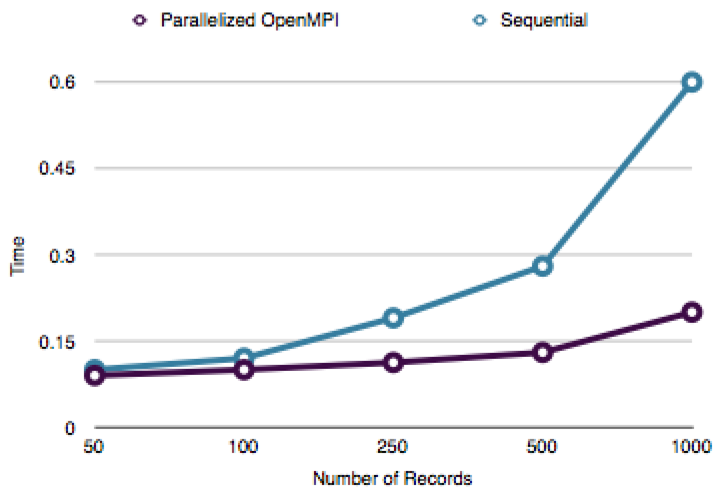
5. System Requirements and Software Dependencies

Our project should be fully runnable on any computer with an updated version of Java. It has been tested and shown to run on the Andrew Linux machines. Note that because OpenMPI has only been installed on certain machines, it will not be guaranteed to function on any machines beyond those we have mentioned in our report for testing.

6. Graphs and Discussion of Parallelism vs Sequentiality

This project, and our implementation, shows the power of computing over large data sets in parallel rather than sequentially, by farming out tasks to various processors on different machines. By Amdahl's Law, the more we parallelize our computation, the greater the speedup as shown in the graph below. Also shown in the graphs, we can see that there is a significant difference in the amount of time associated with computing over larger sets of data increases more so in the sequential case than the parallelized version.


Parallelized OpenMPI
20
15
10
5
0
Speed up
1
2
4
8
16
32
64
128
Number of Nodes


Parallelized OpenMPI
Sequential
0.6
0.45
0.3
0.15
0
Time
50
100
250
500
1000
Number of Records