# 15640 Project4 Report

# Clustering Data Points and DNA Strands Using MPI

Name1: Jian Wang
AndrewID: jianw3
Name2: Yifan Li
AndrewID: yifanl

# Parallelizing Approach

The sequential version of K-Means is like following:

1. We start by deciding how many clusters we would like to form from our data. We call this value k. The value of k is generally a small integer, such as 2, 3, 4, or 5, but may be larger.
2. Next we select k points to be the centroids of k clusters which at present have no members. The list of centroids can be selected by any method (e.g., randomly from the set of data points). It is usually better to pick centroids that are far apart.
3. We then compute the Euclidean distance (the similarity function with a data set of data points) from each data point to each centroid. A data point is assigned to a cluster such that its distance to that cluster is the smallest among all other distances.
4. After associating every data point with one of k clusters, each centroid is recalculated so as to reflect the true mean of its constituent data points.
5. Steps 3 and 4 are repeated for a number of times (say μ), essentially until the centroids start varying very little.

The paralleled version of K-Means will perform the step 1 and step 2 in the sequential version. However, it will split the input raw data into different data splits and dispatch the split to the different processes and run the step 3 in sequential version. On each process, the step 3 result is sent back to a master host, in which place, the clusters are merged together and recalculate a new centroid. The new centroid is sent back to each host and repeat the step 3 again. The period between dispatching the split until sending the new centroids to each host is considered as an iteration. Repeat the iteration for a fixed iterations, the centroid will converge to some reasonable position.

The paralleled part here is trying to split the raw data and run the step 3 in a distributed manner.

# Deployment

Sequential version algorithm and data generator is very straightforward. Please refer to the code.
Require: JRE Java 1.7, Linux, AFS file system, Clusters

1. “cd” to the src/ directory
2. “make”
3. cd ../
4. generate data use “python ./randomclustergen/DNAStrandGen.py -c <cluster

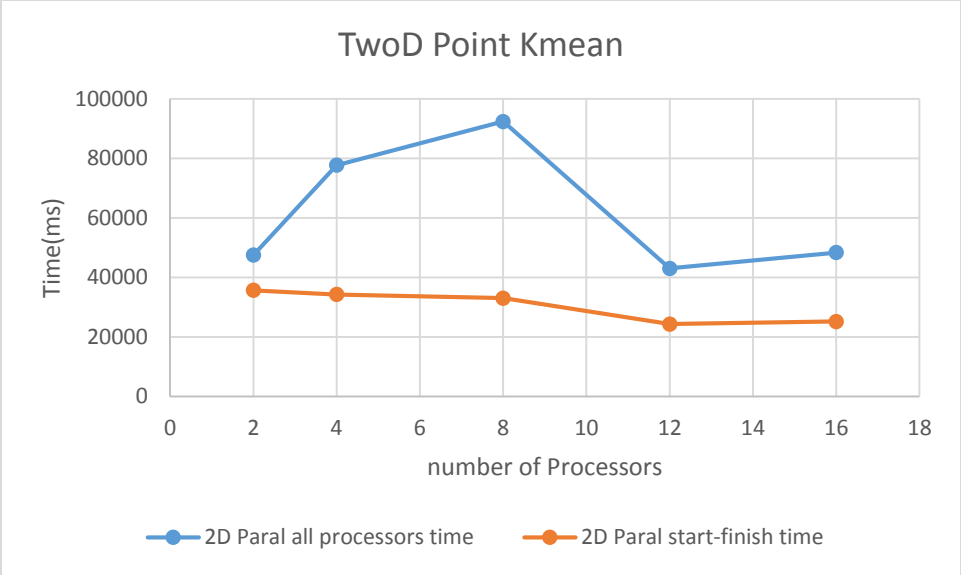
number> -p <data number> -o input/DNA.txt"
for example:
python ./randomclustergen/DNAStrandGen.py -c 5 -p 500000 -o input/DNA.txt

5. "cd" to the **bin** directory
6. For sequential 2D version:
   Type in: **java sequencial.SeqKmeansFor2Dpoints <k> <miu>** in terminal
   Example: **java sequencial.SeqKmeansFor2Dpoints 5 100**
7. For sequential DNA version:
   Type in: **java sequencial.SeqKmeansForDNAStrands <k> <miu>** in terminal
   Example: **java sequencial.SeqKmeansForDNAStrands 5 100**
8. For parallel 2D version:
   Type in: **mpirun -np <number of process> -hostfile ../host_file java parallel.ParallelTwoD <k> <miu>** in terminal
   Example: **mpirun -np 5 -hostfile ../host_file java parallel.ParallelTwoD 5 100**
9. For parallel DNA Strand version:
   Type in: **mpirun -np <number of process> -hostfile ../host_file java parallel.ParallelDNA <k> <miu>** in terminal
   Example: **mpirun -np 5 -hostfile ../host_file java parallel.ParallelDNA 5 100**

10. see the output file in output folder

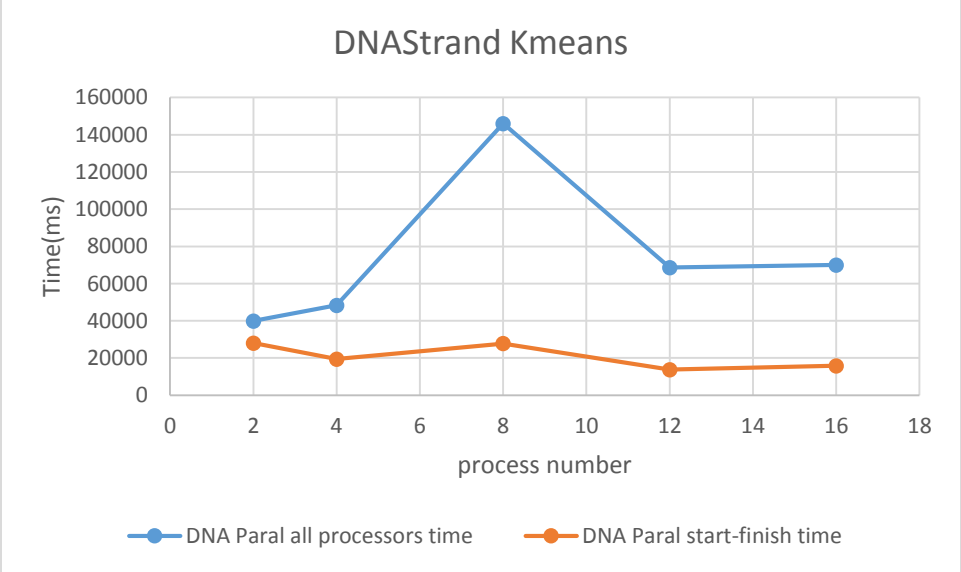
# Experimentation and Analysis

For Two D points experimentation, the dataset size is 500,000,000 points. The iteration number miu is 100, cluster number is 5. See the results below:




The Sequential version of the TwoD points Kmeans is 39351 ms. All the parallel versions are faster than the sequential version.
One thing to note is that when number of processes is 12 and 16, we use 3 hosts(each has 4 cores) to run the experimentation. As we can see, 12 processes is kind of the sweet spot for 3 hosts in our setting.

For DNA Strands experimentation, the dataset size 10,000 points. The iteration number miu is 100, cluster number is 5. See the results below:




The Sequential version of the DNA points Kmeans is 17954 ms. When process number is 12 , the time is smaller. If we test larger data, parallel version is better.