K-Means Clustering Using MPI

CPSC 479 Intro to High Performance Computing
Alex Ho
Chandler Ebrahimi
Darren Vu

Project Goals

Sometimes we just want to see how data is organized, and that's where clustering comes into play. The world 'clustering' means grouping similar things together.

In this project we are interested in the most commonly used clustering method, K-Means. We implement K-Means with the open-source library MPI (Message Passing Interface) to apply high performance computing techniques in data science.

Implementation

1. Create random data point numbers from 0 to 1, and assign partial number points to each processor. (MPI_Scatter)
2. Choose the first few data points as *centroids* and assign them to cluster.
3. In each processor for each data point find it's cluster by calculating it's distance with centroids. (MPI_bcast for centroid list and mean distance tracking if mean reach equally among centroids)
4. Calculate the mean distance of each cluster, and update centroids for each cluster. (MPI_Reduce to get points distance from each process)
5. Repeat Step 3 until the number of iterations is greater than 10,000 or mean distance has changed less than 0.00001.
6. Label all points with it's cluster and print the results

Executing the program

Use mpicc to compile main.c (mpicc main.c), then run the compile file with four input arguments.

mpirun -n <number process> a.out <k number or number of clusters> <number dimension> <number points>

Example: **mpicc main.c && mpirun -n 6 a.out 2 2 100**

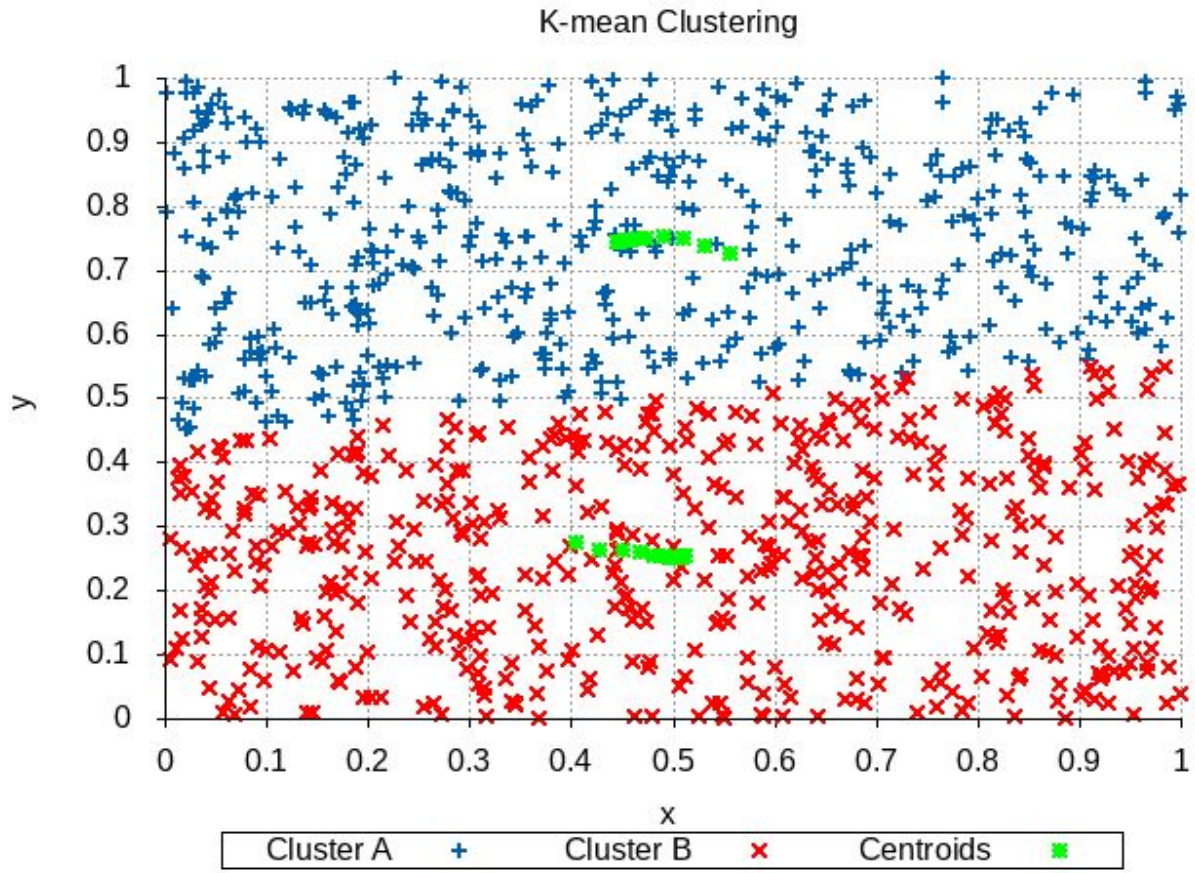
This will use 6 processes with 2 K-Means and 2 dimensions for 100 data points.

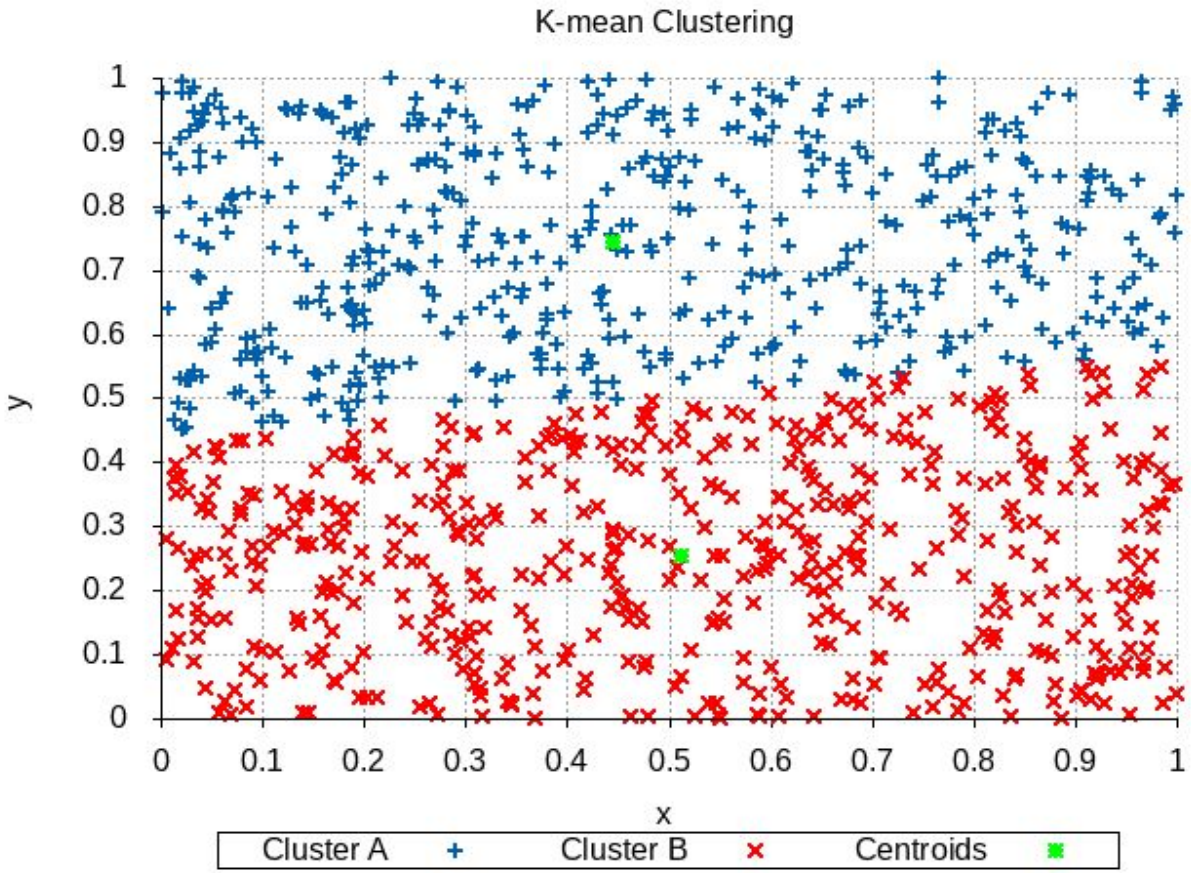
Outputting Images

Output images require ***gnuplot*** and will only work for 2 dimensions and 3 max K number.

Create K-Means cluster with 2 clusters and 100 data points per process:

mpicc main.c && mpirun -n 10 a.out 2 2 1000 && gnuplot graphics/2_kmean_graph.gp


K-mean Clustering
y
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Cluster A
Cluster B
Centroids

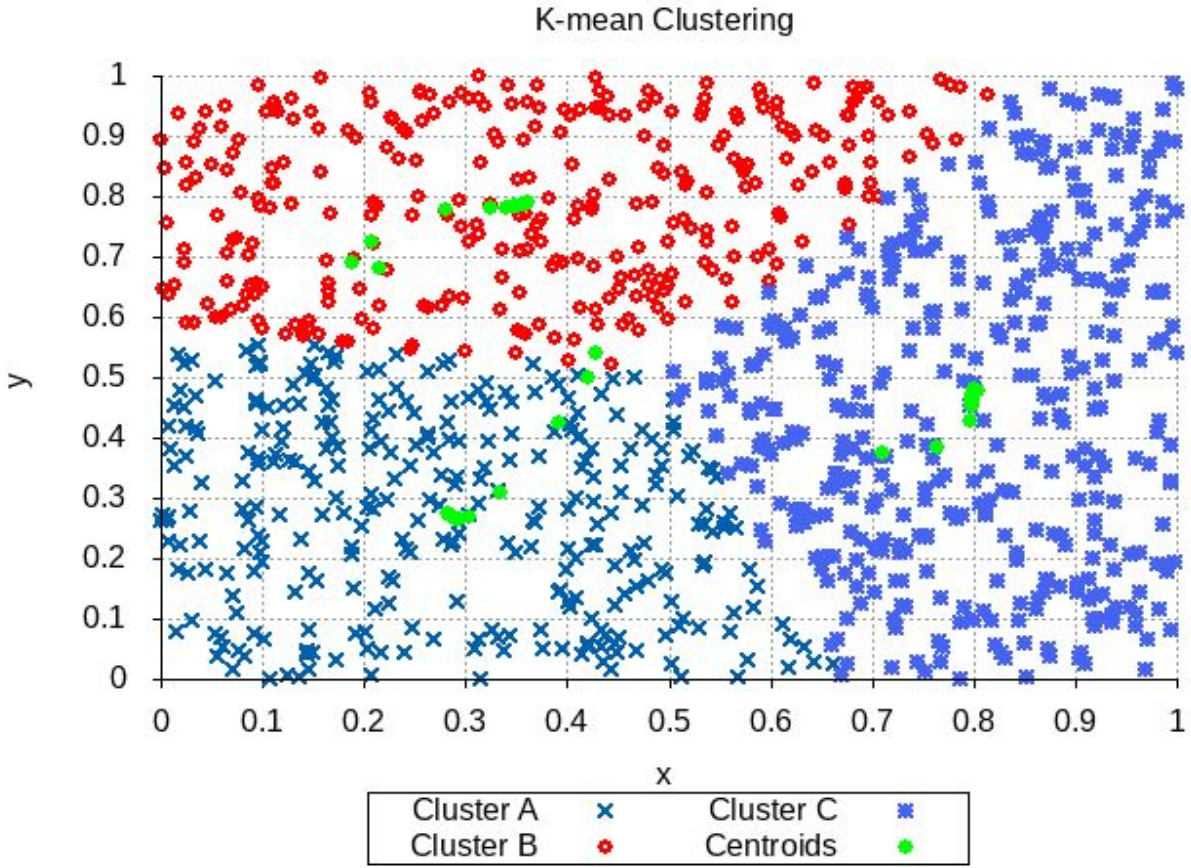

K-mean Clustering
y
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Cluster A
Cluster B
Centroids

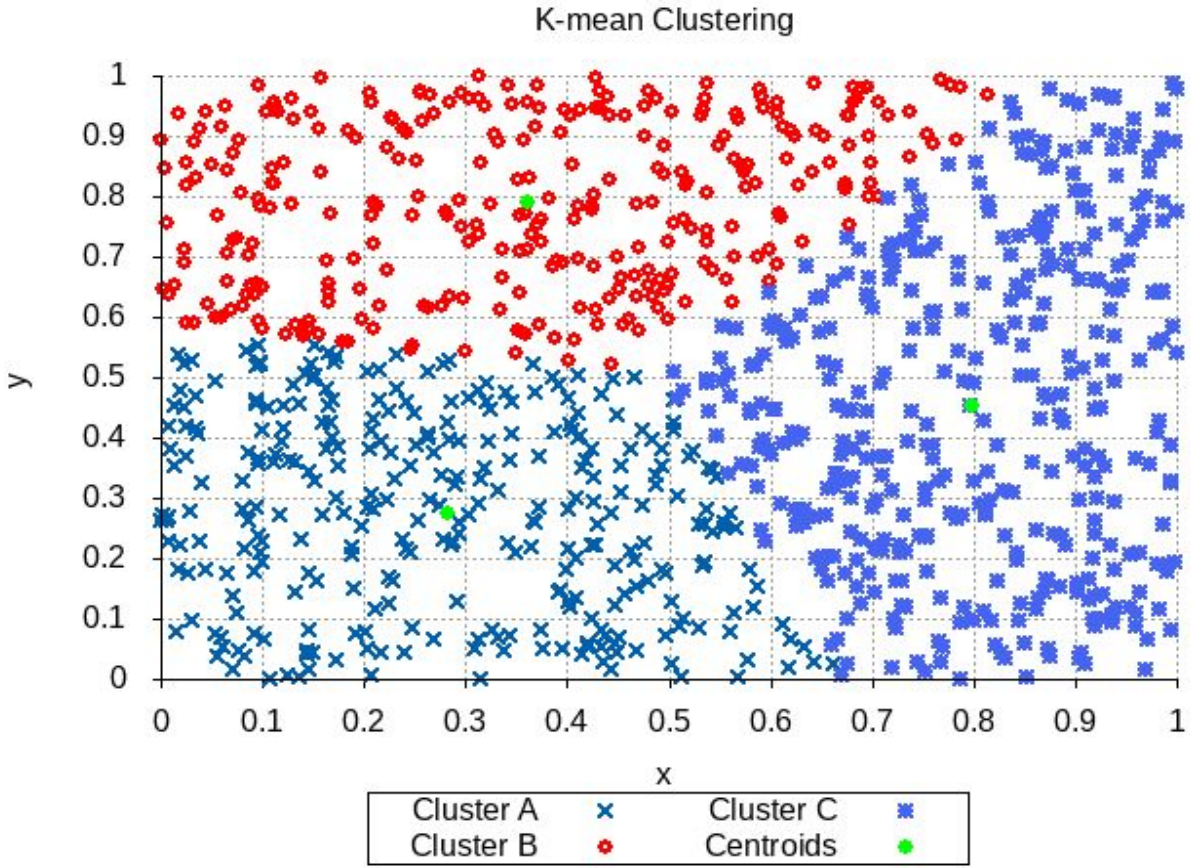
Outputting Images cont.

This time we will compute 3 K-Means clustering for 2 dimensions with a graph.

mpicc main.c && mpirun -n 10 a.out 3 2 1000 && gnuplot graphs/3_kmean_graph.gp

Alternative: run make

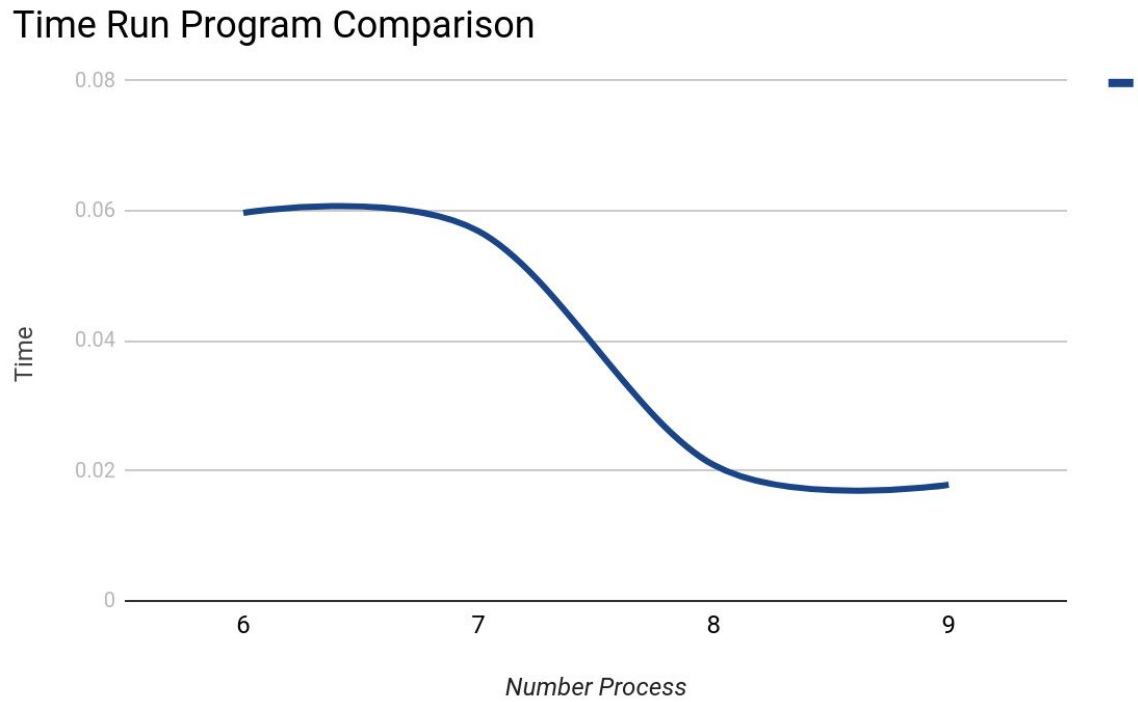

K-mean Clustering
y
x
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Cluster A
Cluster B
Cluster C
Centroids


K-mean Clustering
y
x
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Cluster A
Cluster B
Cluster C
Centroids

Time table

This table illustrates the run time over an increasing number of processes. K-number is 2, dimensions are 2, and the total points is 1,000.

N	6	7	8	9
Time(ms)	0.059706	0.056879	0.03020904	0.017869



Conclusion: The more processes we used to cluster, the faster the program will be.

Difficulties

Figure which MPI function to use.

Divide work among processes.

Setup MPI on IOS. (Current only work with mpi linux)

Code and Demo

Questions