Overview

The assignment work is wrapped up in a zip file. It contains four folders in the root directory: **act1** is for programming assignment #1 and **plot** is for testing with python plot. Also, there is a file called **a5_table_calculations.pdf** in the root folder which contains the raw calculation of data for tables.

Q1: What component of the algorithm is straightforward to parallelize?
Ans: The assignment of all points in the dataset to the nearest centroid using Euclidean distance is straightforward to parallelize.

Q1: When parallelizing the algorithm, what component of the algorithm requires communication between process ranks?
Ans: The component where the next location of centroid is updated for the next iteration requires communication between process ranks. The communication is required for calculation of weighted centroid and updating new centroids to all ranks.

Programming Activity #1

All the files for program activity 1 are inside the **act1** folder.

Files

1. **kmeans_act1_mp2525.c**
   The main C file for programming activity 1.
2. **jobscript_p_<number_of_processors>_k_<number_of_k_means>.sh**
   The job script file is formatted in terms of processor numbers and number of clusters (k).

Main Program

1. The dataset is imported and assigned in a *dataset* variable
2. Start the local total time
3. Calculate the ranges of data and scatter to all ranks from rank 0
    a. Declare startRanges and endRanges array
    b. Rank 0 calculates start and end range for all ranks
    c. If there are leftovers data, It is assigned to last rank
    d. The range values are scattered to respective ranks
4. Allocate memory and initialize centroids with first *KMEANS* points in dataset
5. Declare local and global cluster counter array initializing with 0 value
6. Declare *partialMean* and *localClusterCount*
7. Iterate the kmean algorithm for *KMEANITERS* times
    a. Assign *partialMean* with 0.0 value
    b. Assign *localClusterCount* with 0 value
    c. Start the time for distance calculation
    d. Calculate distance between centroids and points in dataset associated to rank
    e. Find minDistance and assign point with cluster (where the centroid point is located)
    f. Count the number of points in a localcluster
    g. Stop the time for distance calculation and Start time for centroid update
    h. Add distance calculation time to local cumulative distance calculation time
    i. The local cluster count is reduced with sum function and sent to all ranks
    j. Calculate partial mean from dataset value and cluster count value for each clusters
    k. The partial mean is reduced with sum function as new centroid value and sent to all ranks
    l. Stop the centroid update time
    m. Add centroid update time to local cumulative centroid update time
8. Stop the local total time
9. Calculate local total time
10. Reduce into the max value of local distance calculation time to all ranks
11. Reduce into the max value of local centroid update time to all ranks
12. Reduce into the max value of local total time to all ranks
13. Print the maximum total time from rank 0
14. Print the global distance calculation, centroid update and total time
15. Free memory allocations for *centroids, clusters, globalClusterCount, dataset, startRanges* and *endRanges.*

```
[mp2525@rain ~/CS599-HPC/k-means/act1 ]$ sbatch jobscript_p_4_k_2.sh
Submitted batch job 29428949
[mp2525@rain ~/CS599-HPC/k-means/act1 ]$ cat /scratch/mp2525/kmeans_act1_mp2525_p4_k2.txt

Number of lines (N): 5159737, Dimensionality: 2, KMEANS: 2, Filename: ../iono_57min_5.16Mpts_2D.txt
#######################################
Distance calculation time: 0.952018
Centroid update time: 0.071279
TOTAL time taken: 1.030683
#######################################
[mp2525@rain ~/CS599-HPC/k-means/act1 ]$
```

Fig: Running the jobscript in assignment #1 with nodes = 1, p = 4, and k = 2

```
[mp2525@rain ~/CS599-HPC/k-means/act1 ]$ sbatch jobscript_p_16_k_50.sh
Submitted batch job 29429037
[mp2525@rain ~/CS599-HPC/k-means/act1 ]$ cat /scratch/mp2525/kmeans_act1_mp2525_p16_k50.txt

Number of lines (N): 5159737, Dimensionality: 2, KMEANS: 50, Filename: ../iono_57min_5.16Mpts_2D.txt
#######################################
Distance calculation time: 1.840109
Centroid update time: 0.074229
TOTAL time taken: 1.923491
#######################################
[mp2525@rain ~/CS599-HPC/k-means/act1 ]$
```

Fig: Running the jobscript in assignment #1 with nodes = 1, p = 16, and k = 50

```
[mp2525@rain ~/CS599-HPC/k-means/act1 ]$ sbatch jobscript_p_20_k_100.sh
Submitted batch job 29429038
[mp2525@rain ~/CS599-HPC/k-means/act1 ]$ cat /scratch/mp2525/kmeans_act1_mp2525_p20_k100.txt

Number of lines (N): 5159737, Dimensionality: 2, KMEANS: 100, Filename: ../iono_57min_5.16Mpts_2D.txt
#######################################
Distance calculation time: 2.806127
Centroid update time: 0.107321
TOTAL time taken: 2.924931
#######################################
[mp2525@rain ~/CS599-HPC/k-means/act1 ]$
```

Fig: Running the jobscript in assignment #1 with nodes = 1, p = 20, and k = 100

Table 1: Total response time.

# of Ranks (p)	k=2	k=25	k=50	k=100	Job Script Name (*.sh)
1	3.452691	14.69657267	27.529456	52.68849067	jobscript_p_1_k_[2-100].sh
4	0.873296	3.984103667	7.251848	13.88945667	jobscript_p_4_k_[2-100].sh
8	0.4583223333	1.979737667	3.701249667	7.232404667	jobscript_p_8_k_[2-100].sh
12	0.3247543333	1.340422667	2.474371333	4.798564	jobscript_p_12_k_[2-100].sh
16	0.2501896667	1.011012333	1.880746333	3.666174667	jobscript_p_16_k_[2-100].sh
20	0.19846	0.8322896667	1.532935333	2.938806667	jobscript_p_20_k_[2-100].sh

Table 2: Maximum cumulative time spent performing distance calculations.

# of Ranks (p)	k=2	k=25	k=50	k=100	Job Script Name (*.sh)
1	3.158562	14.04557133	26.398908	50.67164833	jobscript_p_1_k_[2-100].sh
4	0.8005033333	3.814786667	6.960614667	13.374825	jobscript_p_4_k_[2-100].sh
8	0.4146446667	1.896434667	3.548503333	6.954961333	jobscript_p_8_k_[2-100].sh
12	0.284245	1.274192	2.367189	4.611627667	jobscript_p_12_k_[2-100].sh
16	0.2134296667	0.9589296667	1.798057667	3.513603667	jobscript_p_16_k_[2-100].sh
20	0.1676583333	0.7868203333	1.462909	2.819595333	jobscript_p_20_k_[2-100].sh

Table 3: Maximum cumulative time spent updating means, including performing synchronization between ranks.

# of Ranks (p)	k=2	k=25	k=50	k=100	Job Script Name (*.sh)
1	0.2878156667	0.64465	1.1241233333	2.010416667	jobscript_p_1_k_[2-100].sh
4	0.064856	0.16221	0.2846103333	0.508014333	jobscript_p_4_k_[2-100].sh
8	0.038409	0.0849136666	0.1447713333	0.259037666	jobscript_p_8_k_[2-100].sh
12	0.0325923333	0.0605336666	0.0983283333	0.178361	jobscript_p_12_k_[2-100].sh
16	0.0274323333	0.0431456666	0.073903	0.143662666	jobscript_p_16_k_[2-100].sh
20	0.020618333	0.0350953333	0.0600443333	0.108602	jobscript_p_20_k_[2-100].sh

Table 4: Speedup computed using the data in Table 1.

# of Ranks (p)	k=2	k=25	k=50	k=100
1	1	1	1	1
4	3.953631987	3.688802776	3.796198707	3.793416253
8	7.5333248	7.423494998	7.437881386	7.285058441
12	10.63170109	10.9641332	11.12583856	10.98005375
16	13.80029418	14.53649197	14.63751677	14.37151676
20	17.3974151	17.65800208	17.95865449	17.92853244

Q3: Describe how you implemented your algorithm.
Ans: In the implementation of my algorithm, first, rank 0 calculates data ranges for all the ranks and scatter to respective ranks. The K-mean algorithm is implemented in ten iterations. The distance is calculated from k centroids to points in a rank within their respective range. The point with minimum distance with centroid is assigned with the cluster associated with centroid point. The centroid update is done by a weighted mean method (Idea 2) where the total count of the cluster is collected and divided with a partial sum of points. The newly centroid value is updated to all the ranks using all reduce function with sum operation.

Q4: Explain the (potential) bottlenecks in your algorithm.
Ans: There is a communication overhead when updating the centroids points. We need to communicate two times during this process in each iteration.

Q5: How does the algorithm scale when k=2? Explain.
Ans: The algorithm scales well when k = 2. The speedup tends to improve with the increasing number of processors improving the distance calculation, centroid update and total time.

Q6: How does the algorithm scale when k=100? Explain.
Ans: The algorithm scales well when k = 100. The speedup tends to improve with the increasing number of processors significantly improving the distance calculation, centroid update and total time. The elapsed time is heavily improved scaling from one processor to multi processors.

K-Means: Multiple Nodes

```
[mp2525@rain ~ ]$ cd CS599-HPC/k-means/act1-2nodes
[mp2525@rain ~/CS599-HPC/k-means/act1-2nodes ]$ sbatch jobscript_p_24_k_25.sh
Submitted batch job 29480905
[mp2525@rain ~/CS599-HPC/k-means/act1-2nodes ]$ cat /scratch/mp2525/kmeans_act1_mp2525_p24_k25.txt

Number of lines (N): 5159737, Dimensionality: 2, KMEANS: 25, Filename: ../iono_57min_5.16Mpts_2D.txt
#######################################
Distance calculation time: 0.646366
Centroid update time: 0.037227
TOTAL time taken: 0.714005
#######################################
[mp2525@rain ~/CS599-HPC/k-means/act1-2nodes ]$
```

Fig: Running the jobscript in assignment #1 with nodes = 2, p = 24, and k = 25

```
[mp2525@rain ~/CS599-HPC/k-means/act1-2nodes ]$ sbatch jobscript_p_28_k_2.sh
Submitted batch job 29481597
[mp2525@rain ~/CS599-HPC/k-means/act1-2nodes ]$ cat /scratch/mp2525/kmeans_act1_mp2525_p28_k2.txt

Number of lines (N): 5159737, Dimensionality: 2, KMEANS: 2, Filename: ../iono_57min_5.16Mpts_2D.txt
#######################################
Distance calculation time: 0.120192
Centroid update time: 0.014701
TOTAL time taken: 0.155953
#######################################
[mp2525@rain ~/CS599-HPC/k-means/act1-2nodes ]$
```

Fig: Running the jobscript in assignment #1 with nodes = 2, p = 28, and k = 2

```
[mp2525@rain ~/CS599-HPC/k-means/act1-2nodes ]$ sbatch jobscript_p_36_k_50.sh
Submitted batch job 29481598
[mp2525@rain ~/CS599-HPC/k-means/act1-2nodes ]$ cat /scratch/mp2525/kmeans_act1_mp2525_p36_k50.txt

Number of lines (N): 5159737, Dimensionality: 2, KMEANS: 50, Filename: ../iono_57min_5.16Mpts_2D.txt
#######################################
Distance calculation time: 0.812816
Centroid update time: 0.048745
TOTAL time taken: 0.888869
#######################################
[mp2525@rain ~/CS599-HPC/k-means/act1-2nodes ]$
```

Fig: Running the jobscript in assignment #1 with nodes = 2, p = 36, and k = 50

Table 5: Total response time.

# of Ranks (p)	k=2	k=25	k=50	k=100	Job Script Name (*.sh)
24	0.1846933333	0.7205503333	1.320854667	2.461894667	jobscript_p_24_k_[2-100].sh
28	0.165928	0.6185826667	1.129251667	2.158633667	jobscript_p_28_k_[2-100].sh
32	0.1605966667	0.5696903333	1.001307333	1.9074	jobscript_p_32_k_[2-100].sh
36	0.1507946667	0.5424966667	0.934515	1.743406333	jobscript_p_36_k_[2-100].sh
40	0.1442276667	0.490739	0.8345116667	1.553463667	jobscript_p_40_k_[2-100].sh

Table 6: Maximum cumulative time spent performing distance calculations.

# of Ranks (p)	k=2	k=25	k=50	k=100	Job Script Name (*.sh)
24	0.150443	0.6501053333	1.234645333	2.334853	jobscript_p_24_k_[2-100].sh
28	0.1198976667	0.559136	1.051416667	2.035982333	jobscript_p_28_k_[2-100].sh
32	0.1219906667	0.5057566667	0.9300576667	1.807197	jobscript_p_32_k_[2-100].sh
36	0.0987156666	0.4374423333	0.8276993333	1.580649667	jobscript_p_36_k_[2-100].sh
40	0.088441	0.3963466667	0.7416926667	1.424306667	jobscript_p_40_k_[2-100].sh

Table 7: Maximum cumulative time spent updating means, including performing synchronization between ranks.

# of Ranks (p)	k=2	k=25	k=50	k=100	Job Script Name (*.sh)
24	0.01935	0.037335	0.0598683333	0.103077	jobscript_p_24_k_[2-100].sh
28	0.015929	0.0295713333	0.0488823333	0.094822666	jobscript_p_28_k_[2-100].sh
32	0.0175636666	0.028865	0.0582253333	0.083970333	jobscript_p_32_k_[2-100].sh
36	0.0252776666	0.0706816666	0.077783	0.136681333	jobscript_p_36_k_[2-100].sh
40	0.02498	0.0625833333	0.0570863333	0.102663333	jobscript_p_40_k_[2-100].sh

Table 8: Speedup computed using the data in Table 1.

# of Ranks (p)	k=2	k=25	k=50	k=100
24	18.69418315	20.39631652	20.84215372	21.40160234
28	20.80836869	23.75846182	24.3784949	24.40825948
32	21.49914486	25.79747594	27.49351281	27.62319947
36	22.8966387	27.09062299	29.45854909	30.2215781
40	23.93917256	29.94783922	32.98870118	33.91678338

Q7: On two nodes, how does the algorithm scale when K=2? Compare with the single node results.
Ans: The algorithm scales well on two nodes when k = 2 compared to a single node. However, the speedup tends to scale slowly when the number of ranks is increased from 24 to 40.

Q8: On two nodes, how does the algorithm scale when K=100? Compare with the single node results.
Ans: The algorithm scales well on two nodes when k = 100 compared to a single node. Unlike k = 2, the speedup increases highly with increasing p in this algorithm with two nodes. However, the speedup is highly contributed by improvement in the distance calculation time. The mean update time seems to be bottleneck because of communications in k=100. It is because the mean update time is better when k = 50, 25 and 2 compared to k =100.

Q9: Under what conditions do you expect the k-means algorithm to perform well on multiple nodes (e.g., two or more)? When preparing your response, consider the following factors: the number of centroids, the size of the dataset, the data dimensionality, and the number of iterations.
Ans: I expect the k-mean algorithm to perform well on multiple nodes on the following factors:

- High data dimensionality
- The number of centroids in medium range such as 25 and 50
- High number of datasize
- High number of iteration

The performance of the algorithm starts to degrade once the centroid update time exceeds distance calculation time. When the data dimensionality, dataset size and iteration is increased, then the centroid update time will start to decrease and distance calculation time will increase while being executed in a high parallel way.