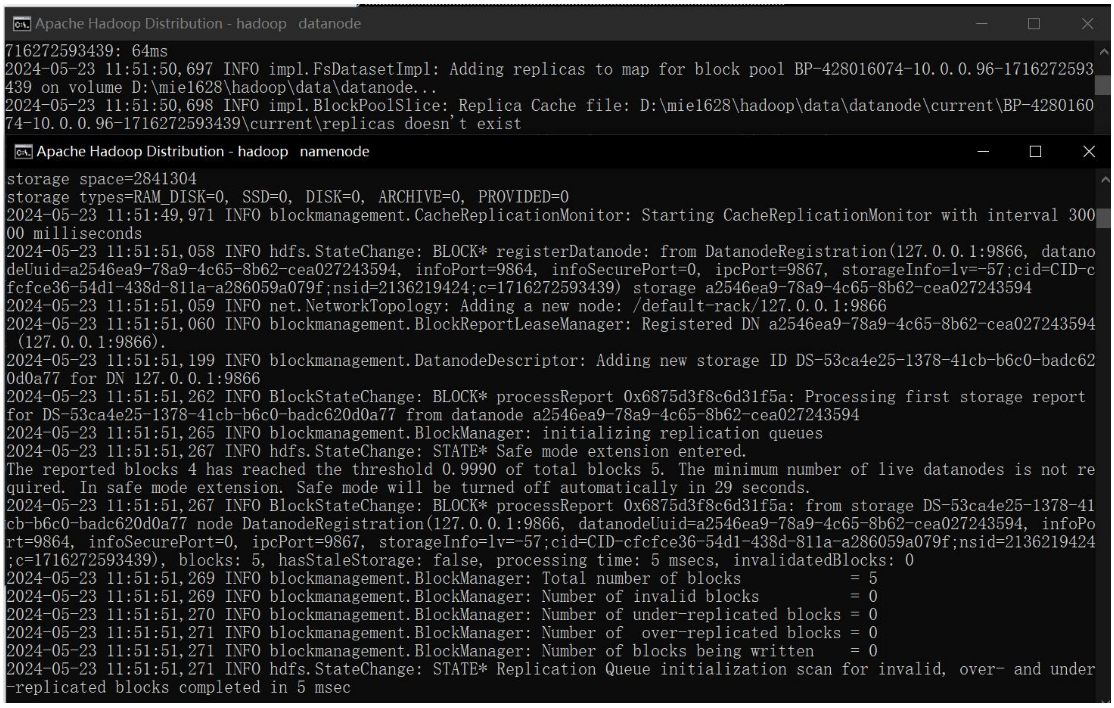
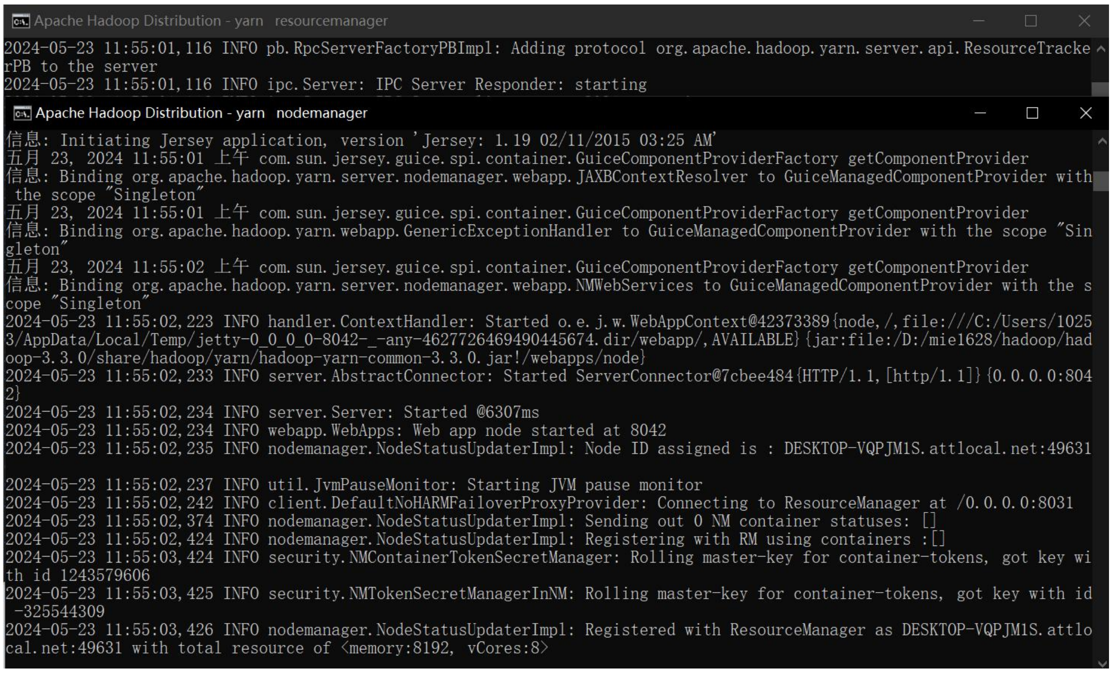
1) [Marks: 15] Implement a Map Reduce program for counting the number of lines in a document.

- Start HDFS daemons



- Start YARN daemons



- Verify Java processes

```
C:\WINDOWS\system32>%HADOOP_HOME%\sbin\start-dfs.cmd

C:\WINDOWS\system32>%HADOOP_HOME%\sbin\start-yarn.cmd
starting yarn daemons

C:\WINDOWS\system32>jps
23456 Jps
10340 NameNode
22868 NodeManager
26420 ResourceManager
24204 DataNode

C:\WINDOWS\system32>
```

- Upload files

```
C:\WINDOWS\system32>hdfs dfs -put /D:/mie1628/hw/A1/shakespeare.txt /Demo

C:\WINDOWS\system32>hdfs dfs -ls /Demo
Found 1 items
-rw-r--r--   1 10253 supergroup    2555806 2024-05-21 02:31 /Demo/shakespeare.txt
```

- Run linecount. jar file

```
C:\WINDOWS\system32>hadoop jar /D:/mie1628/hw/A1/linecount.jar LineCount /Demo/shakespeare.txt /output
2024-05-21 02:35:18,014 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-05-21 02:35:18,826 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the To
ol interface and execute your application with ToolRunner to remedy this.
2024-05-21 02:35:18,910 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/10253/
.staging/job_1716272881766_0001
2024-05-21 02:35:19,324 INFO input.FileInputFormat: Total input files to process : 1
2024-05-21 02:35:19,730 INFO mapreduce.JobSubmitter: number of splits:1
2024-05-21 02:35:19,975 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1716272881766_0001
2024-05-21 02:35:19,975 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-05-21 02:35:20,109 INFO conf.Configuration: resource-types.xml not found
2024-05-21 02:35:20,109 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-05-21 02:35:20,337 INFO impl.YarnClientImpl: Submitted application application_1716272881766_0001
2024-05-21 02:35:20,374 INFO mapreduce.Job: The url to track the job: http://DESKTOP-VQPJM1S:8088/proxy/application_1716272881
766_0001/
2024-05-21 02:35:20,377 INFO mapreduce.Job: Running job: job_1716272881766_0001
2024-05-21 02:35:29,531 INFO mapreduce.Job: Job job_1716272881766_0001 running in uber mode : false
2024-05-21 02:35:29,531 INFO mapreduce.Job:  map 0% reduce 0%
2024-05-21 02:35:34,640 INFO mapreduce.Job:  map 100% reduce 0%
2024-05-21 02:35:41,716 INFO mapreduce.Job:  map 100% reduce 100%
2024-05-21 02:35:42,735 INFO mapreduce.Job: Job job_1716272881766_0001 completed successfully
2024-05-21 02:35:42,840 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=17
                FILE: Number of bytes written=530671
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=2555913
                HDFS: Number of bytes written=11
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=3341
                Total time spent by all reduces in occupied slots (ms)=4108
                Total time spent by all map tasks (ms)=3341
                Total time spent by all reduce tasks (ms)=4108
                Total vcore-milliseconds taken by all map tasks=3341
                Total vcore-milliseconds taken by all reduce tasks=4108
                Total megabyte-milliseconds taken by all map tasks=3421184
                Total megabyte-milliseconds taken by all reduce tasks=4206592
```

```
	Map-Reduce Framework
		Map input records=58483
		Map output records=58483
		Map output bytes=526347
		Map output materialized bytes=17
		Input split bytes=107
		Combine input records=58483
		Combine output records=1
		Reduce input groups=1
		Reduce shuffle bytes=17
		Reduce input records=1
		Reduce output records=1
		Spilled Records=2
		Shuffled Maps =1
		Failed Shuffles=0
		Merged Map outputs=1
		GC time elapsed (ms)=59
		CPU time spent (ms)=1295
		Physical memory (bytes) snapshot=546697216
		Virtual memory (bytes) snapshot=794750976
		Total committed heap usage (bytes)=412614656
		Peak Map Physical memory (bytes)=322359296
		Peak Map Virtual memory (bytes)=396488704
		Peak Reduce Physical memory (bytes)=224337920
		Peak Reduce Virtual memory (bytes)=401231872
	Shuffle Errors
		BAD_ID=0
		CONNECTION=0
		IO_ERROR=0
		WRONG_LENGTH=0
		WRONG_MAP=0
		WRONG_REDUCE=0
	File Input Format Counters
		Bytes Read=2555806
	File Output Format Counters
		Bytes Written=11
```

- Get output

```
C:\WINDOWS\system32>hdfs dfs -ls /output
Found 2 items
-rw-r--r--   1 10253 supergroup          0 2024-05-21 02:35 /output/_SUCCESS
-rw-r--r--   1 10253 supergroup         11 2024-05-21 02:35 /output/part-r-00000

C:\WINDOWS\system32>hdfs dfs -cat /output/part-r-00000
line    58483
```

2) [Marks: 45] Apply K-means clustering on Map Reduce using k = 5 and k = 8 clusters on the given dataset, list the cluster labels or centroids, the number of iterations for convergence or use maximum iterations = 15 and time/duration.

- Generate k initial centers of mass

```
def generate_initial_centroids(input_file, output_file, k):
    with open(input_file, 'r') as f:
        lines = f.readlines()

    centroids = random.sample(lines, k)

    with open(output_file, 'w') as f:
        for centroid in centroids:
            f.write(centroid)

# Generate k initial centers of mass
generate_initial_centroids('D:\mie1628\hw\A2\data_points.txt', 'initial_centroids_8.txt', 8)
```

- Upload Dataset to HDFS:

```
C:\WINDOWS\system32>hdfs dfs -put /D:/mie1628/hw/A2/data_points.txt /Demo

C:\WINDOWS\system32>hdfs -ls /Demo
Unrecognized option: -ls
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.

C:\WINDOWS\system32>hdfs dfs -ls /Demo
Found 2 items
-rw-r--r--   1 10253 supergroup   36938010 2024-05-23 13:35 /Demo/data_points.txt
-rw-r--r--   1 10253 supergroup    2555806 2024-05-21 02:31 /Demo/shakespeare.txt
```

- Upload initial centroids files to HDFS

```
C:\WINDOWS\system32>hdfs dfs -put D:\mie1628\hw\A2\KMeans\initial_centroids_5.txt /Demo

C:\WINDOWS\system32>hdfs dfs -put D:\mie1628\hw\A2\KMeans\initial_centroids_8.txt /Demo

C:\WINDOWS\system32>hdfs dfs -ls /Demo
Found 4 items
-rw-r--r--   1 10253 supergroup   36938010 2024-05-23 13:35 /Demo/data_points.txt
-rw-r--r--   1 10253 supergroup        190 2024-05-23 14:28 /Demo/initial_centroids_5.txt
-rw-r--r--   1 10253 supergroup        305 2024-05-23 14:28 /Demo/initial_centroids_8.txt
-rw-r--r--   1 10253 supergroup    2555806 2024-05-21 02:31 /Demo/shakespeare.txt
```

- Compile to generate jar files:

```
D:\mie1628\hw\A2>cd KMeans

D:\mie1628\hw\A2\KMeans>javac -classpath "D:\mie1628\hadoop\hadoop-3.3.0\share\hadoop\common\lib\*;D:\mie1628\hadoop\hadoop-3.3.0\share\hadoop\common\*;D:\mie1628\hadoop\hadoop-3.3.0\share\hadoop\hdfs\lib\*;D:\mie1628\hadoop\hadoop-3.3.0\share\hadoop\hdfs\*;D:\mie1628\hadoop\hadoop-3.3.0\share\hadoop\yarn\lib\*;D:\mie1628\hadoop\hadoop-3.3.0\share\hadoop\yarn\*;D:\mie1628\hadoop\hadoop-3.3.0\share\hadoop\mapreduce\*" -d classes src/KMeansMapper.java src/KMeansReducer.java src/KMeansDriver.java

D:\mie1628\hw\A2\KMeans>jar -cvf KMeans.jar -C classes .
已添加清单
正在添加: KMeansDriver.class(输入 = 4345) (输出 = 2229)(压缩了 48%)
正在添加: KMeansMapper.class(输入 = 2817) (输出 = 1277)(压缩了 54%)
正在添加: KMeansReducer.class(输入 = 2790) (输出 = 1308)(压缩了 53%)

D:\mie1628\hw\A2\KMeans>
```

- Submitting MapReduce Jobs to a Hadoop Cluster

K = 5:

```
C:\WINDOWS\system32>hadoop jar /D:/mie1628/hw/A2/KMeans/KMeans.jar KMeansDriver /Demo/data_points.txt /Demo/output /Demo/initial_centroids_5.txt 5
2024-05-23 14:40:06,015 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-05-23 14:40:06,593 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your ap
plication with ToolRunner to remedy this.
2024-05-23 14:40:06,754 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/10253/.staging/job_1716479700522_0006
2024-05-23 14:40:07,072 INFO input.FileInputFormat: Total input files to process : 1
2024-05-23 14:40:07,487 INFO mapreduce.JobSubmitter: number of splits:1
2024-05-23 14:40:07,723 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1716479700522_0006
2024-05-23 14:40:07,723 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-05-23 14:40:07,842 INFO conf.Configuration: resource-types.xml not found
2024-05-23 14:40:07,843 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-05-23 14:40:07,891 INFO impl.YarnClientImpl: Submitted application application_1716479700522_0006
2024-05-23 14:40:07,927 INFO mapreduce.Job: The url to track the job: http://DESKTOP-VQPJM1S:8088/proxy/application_1716479700522_0006/
2024-05-23 14:40:07,928 INFO mapreduce.Job: Running job: job_1716479700522_0006
2024-05-23 14:40:15,063 INFO mapreduce.Job: Job job_1716479700522_0006 running in uber mode : false
2024-05-23 14:40:15,064 INFO mapreduce.Job:  map 0% reduce 0%
2024-05-23 14:40:23,186 INFO mapreduce.Job:  map 100% reduce 0%
2024-05-23 14:40:32,272 INFO mapreduce.Job:  map 100% reduce 100%
2024-05-23 14:40:33,290 INFO mapreduce.Job: Job job_1716479700522_0006 completed successfully
2024-05-23 14:40:33,370 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=42938017
                FILE: Number of bytes written=86406933
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=36938117
                HDFS: Number of bytes written=197
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
```

```
                Total vcore-milliseconds taken by all map tasks=6180
                Total vcore-milliseconds taken by all reduce tasks=6956
                Total megabyte-milliseconds taken by all map tasks=6328320
                Total megabyte-milliseconds taken by all reduce tasks=7122944
        Map-Reduce Framework
                Map input records=1000000
                Map output records=1000000
                Map output bytes=40938011
                Map output materialized bytes=42938017
                Input split bytes=107
                Combine input records=0
                Combine output records=0
                Reduce input groups=5
                Reduce shuffle bytes=42938017
                Reduce input records=1000000
                Reduce output records=5
                Spilled Records=2000000
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=177
                CPU time spent (ms)=6406
                Physical memory (bytes) snapshot=836399104
                Virtual memory (bytes) snapshot=1251074048
                Total committed heap usage (bytes)=898105344
                Peak Map Physical memory (bytes)=396931072
                Peak Map Virtual memory (bytes)=688107520
                Peak Reduce Physical memory (bytes)=439468032
                Peak Reduce Virtual memory (bytes)=562978816
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=36938010
        File Output Format Counters
                Bytes Written=188
Iteration 14 completed in 33513 ms
```

Check the result:

```
C:\WINDOWS\system32>hdfs dfs -cat /Demo/output_0/part-r-00000
0       33.13449282411685,0.5243838915790391
1       14.518114067298372,14.618519626843364
2       49.54106083199774,32.02260432776371
3       51.97211218048171,23.704287378994767
4       43.242099428615724,-10.770157466970533

C:\WINDOWS\system32>hdfs dfs -cat /Demo/output_14/part-r-00000
0       34.9656892655878,4.171051386474803
1       9.92750128502909,15.118838492653879
2       49.958740637143066,33.0095498127739
3       50.03271829114198,24.925013331182246
4       34.75848121228197,-4.328259665154
```

K = 8:

```
C:\WINDOWS\system32>hadoop jar /D:/mie1628/hw/A2/KMeans/KMeans.jar KMeansDriver /Demo/data_points.txt /Demo/output_8 /Demo/initial_centroids_8.txt 8
2024-05-23 14:53:23,064 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-05-23 14:53:23,648 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed.  Implement the Tool interface and execute your ap
plication with ToolRunner to remedy this.
2024-05-23 14:53:23,719 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/10253/.staging/job_1716479700522_0021
2024-05-23 14:53:24,044 INFO input.FileInputFormat: Total input files to process : 1
2024-05-23 14:53:24,440 INFO mapreduce.JobSubmitter: number of splits:1
2024-05-23 14:53:24,667 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1716479700522_0021
2024-05-23 14:53:24,667 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-05-23 14:53:24,798 INFO conf.Configuration: resource-types.xml not found
2024-05-23 14:53:24,799 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-05-23 14:53:24,849 INFO impl.YarnClientImpl: Submitted application application_1716479700522_0021
2024-05-23 14:53:24,882 INFO mapreduce.Job: The url to track the job: http://DESKTOP-VQPJM1S:8088/proxy/application_1716479700522_0021/
2024-05-23 14:53:24,883 INFO mapreduce.Job: Running job: job_1716479700522_0021
2024-05-23 14:53:32,023 INFO mapreduce.Job: Job job_1716479700522_0021 running in uber mode : false
2024-05-23 14:53:32,025 INFO mapreduce.Job:  map 0% reduce 0%
2024-05-23 14:53:39,130 INFO mapreduce.Job:  map 100% reduce 0%
2024-05-23 14:53:48,223 INFO mapreduce.Job:  map 100% reduce 100%
2024-05-23 14:53:49,243 INFO mapreduce.Job: Job job_1716479700522_0021 completed successfully
2024-05-23 14:53:49,354 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=42938017
                FILE: Number of bytes written=86407161
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=36938117
                HDFS: Number of bytes written=311
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=5030
                Total time spent by all reduces in occupied slots (ms)=6218
                Total time spent by all map tasks (ms)=5030
                Total time spent by all reduce tasks (ms)=6218
                Total vcore-milliseconds taken by all map tasks=5030
                Total vcore-milliseconds taken by all reduce tasks=6218
                Total vcore-milliseconds taken by all map tasks=5430
                Total vcore-milliseconds taken by all reduce tasks=7150
                Total megabyte-milliseconds taken by all map tasks=5560320
                Total megabyte-milliseconds taken by all reduce tasks=7321600
        Map-Reduce Framework
                Map input records=1000000
                Map output records=1000000
                Map output bytes=40938011
                Map output materialized bytes=42938017
                Input split bytes=107
                Combine input records=0
                Combine output records=0
                Reduce input groups=8
                Reduce shuffle bytes=42938017
                Reduce input records=1000000
                Reduce output records=8
                Spilled Records=2000000
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=213
                CPU time spent (ms)=7370
                Physical memory (bytes) snapshot=839028736
                Virtual memory (bytes) snapshot=1246101504
                Total committed heap usage (bytes)=880279552
                Peak Map Physical memory (bytes)=392941568
                Peak Map Virtual memory (bytes)=681177088
                Peak Reduce Physical memory (bytes)=446087168
                Peak Reduce Virtual memory (bytes)=564985856
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=36938010
        File Output Format Counters
                Bytes Written=307
Iteration 14 completed in 32082 ms

C:\WINDOWS\system32>
```

Result:

```
C:\WINDOWS\system32>hdfs dfs -cat /Demo/output_8_0/part-r-00000
0       43.174996171788166,10.527897658622217
1       10.62663705880069,18.42858769645474
2       49.843015394376195,33.78473126938813
3       34.61989432768522,1.2243814245667284
4       47.35674246472724,24.137154223582172
5       9.283459409791206,11.030044207300435
6       49.57072529714386,28.88087264274518
7       55.27292206757376,26.854467551410725

C:\WINDOWS\system32>hdfs dfs -cat /Demo/output_8_14/part-r-00000
0       34.87452694527554,4.90318673089756
1       10.00727537665053,18.41831050284055
2       49.87738923663939,38.82110915361992
3       34.97438673532728,-2.5127870283694502
4       43.53077443949456,22.102760964080993
5       9.847863021382812,10.61986787245903
6       50.04194493469359,30.09280764133495
7       54.62422784751054,21.808123431791305
```

3) [Marks: 10] Explain the advantages and disadvantages of using K-Means Clustering with MapReduce.

Advantages:

1. Scalability:

- Large Datasets: MapReduce is designed to handle large-scale data processing. K-Means clustering implemented with MapReduce can scale to very large datasets that wouldn't fit into the memory of a single machine.
- Distributed Computing: By leveraging the distributed computing model of MapReduce, K-Means can be executed in parallel across multiple nodes, significantly speeding up the computation.

2. Efficiency:

- Parallel Processing: MapReduce divides the computation into smaller tasks (maps) that can be processed in parallel, and then aggregates the results (reduces), leading to efficient use of computational resources.
- Fault Tolerance: Hadoop’s MapReduce framework includes fault tolerance, which means if a node fails, the system can recover by reassigning tasks to other nodes, ensuring reliable computation.

3. Ease of Integration:

- Hadoop Ecosystem: K-Means Clustering with MapReduce can easily integrate with other components of the Hadoop ecosystem, such as HDFS for data storage and YARN for resource management.
- Existing Infrastructure: Organizations already using Hadoop can integrate K-Means clustering without needing to invest in new infrastructure.

4. Cost-Effectiveness:

- Commodity Hardware: MapReduce is designed to run on commodity hardware, which can be more cost-effective compared to high-end specialized hardware.

Disadvantages

1. Complexity in Implementation:

- Programming Model: Writing MapReduce jobs requires familiarity with the MapReduce programming model, which can be more complex and less intuitive than other high-level programming paradigms.
- Debugging and Maintenance: Debugging MapReduce jobs can be challenging due to the distributed nature of the computation and the large number of intermediate files generated.

2. Limited Flexibility:

- Algorithm Constraints: K-Means with MapReduce might not be as flexible as other implementations. For instance, certain optimizations or modifications to the K-Means algorithm may be difficult to implement within the MapReduce framework.
- Iterative Process: K-Means is inherently iterative, and each iteration requires a new MapReduce job. This leads to overhead from repeatedly reading and writing data to HDFS, making it less efficient compared to in-memory processing frameworks like Spark.

3. Performance Overhead:

- Disk I/O: MapReduce jobs involve significant disk I/O operations due to the need to write intermediate results to HDFS. This can slow down the performance compared to in-memory processing.

4) [Marks: 10] Can we reduce the number of distance comparisons by applying the Canopy Selection? Which distance metric should we use for the canopy clustering and why?

Yes, the number of distance comparisons can be significantly reduced by applying the Canopy Selection method. The key idea behind the canopy method is to divide the data into overlapping subsets (canopies) using a cheap, approximate distance measure. This initial partitioning step helps in limiting the expensive distance calculations to only those pairs of points that fall within the same canopy. By doing this, the total number of distance comparisons is reduced, leading to a more efficient clustering process.

For the given dataset, the canopy clustering method involves two stages: using a cheap distance metric for the initial partitioning into canopies and an expensive distance metric for the precise clustering within those canopies.

Cheap Distance Metric: Manhattan Distance (L1 Norm):
1. Computational Efficiency: Manhattan distance (sum of absolute differences) is computationally inexpensive compared to Euclidean distance. It can be calculated quickly, making it suitable for the initial canopy formation stage.
2. Effective Approximation: Manhattan distance provides a rough but effective measure of similarity, which is sufficient for grouping points into canopies.

Expensive Distance Metric:Euclidean Distance (L2 Norm):
1. Accuracy: Euclidean distance provides a precise measure of similarity, considering the straight-line distance between points in a high-dimensional space.
2. Common Usage: Euclidean distance is widely used in clustering algorithms like K-means due to its accuracy and effectiveness in high-dimensional spaces.

5) [Marks: 10] Is it possible to apply Canopy Selection on MapReduce? If yes, then explain in words, how would you implement it.

Applying Canopy Selection on MapReduce

Yes, it is possible to apply Canopy Selection using the MapReduce paradigm.

1. Initial Partitioning (Mapper Phase)

In the initial partitioning phase, we use a Map job to calculate the cheap distance metric and assign each data point to multiple canopies.

Mapper Function:

- Input: Each mapper takes a subset of data points.
- Output: The mapper outputs canopy assignments.

1). Load Data: Each mapper reads a subset of data points.

2). Calculate Distances: For each data point, calculate the cheap distance (e.g., Manhattan distance) to a randomly chosen set of initial canopy centers.

3). Assign Canopies: Assign each data point to canopies if the distance is less than a specified threshold T1. Use a second threshold T2 to determine whether the point should be removed from the candidate list for new canopies.

4). Emit Canopy Assignments: The mapper emits key-value pairs where the key is the canopy ID and the value is the data point.

2. Reduce Phase for Canopy Formation

The Reducer aggregates the points assigned to each canopy and outputs the canopies.

Reducer Function:

- Input: Each reducer receives a canopy ID and the corresponding data points.
- Output: The reducer outputs the formed canopies.

1). Aggregate Points: Collect all points assigned to the same canopy.

2). Output Canopies: Write the canopies to the output, ensuring they include all assigned points.

3. Precise Clustering Within Canopies (Second MapReduce Job)

In the precise clustering phase, we use another MapReduce job to perform accurate clustering (e.g., K-means) within each canopy using the expensive distance metric.

Mapper Function:

- Input: Each mapper reads the canopies.
- Output: The mapper outputs the data points along with their canopy ID.

Reducer Function:

- Input: Each reducer receives a canopy ID and the corresponding data points.
- Output: The reducer performs precise clustering within each canopy and outputs the final clusters.

Steps:

1). Load Canopies: Each mapper reads a canopy.

2). Emit Data Points: Each mapper emits the data points along with their canopy ID.

3). Perform Clustering: Each reducer receives a canopy's data points, performs precise clustering using an expensive distance metric (e.g., Euclidean distance), and outputs the clusters.

6) [Marks: 10] Is it possible to combine the Canopy Selection with K-Means on MapReduce? If yes, then explain in words, how would you do that.

Yes, it is possible to combine Canopy Selection with K-Means clustering using the MapReduce framework.

Combining Canopy Selection with K-Means on MapReduce involves two main phases:
1. Canopy Selection Phase: Using a cheap distance metric to form overlapping canopies of data points.
2. K-Means Clustering Phase: Using an expensive distance metric to perform K-Means clustering within each canopy.

Phase 1: Canopy Selection

1. Mapper Function
2. Reducer Function
Both are the same as Q5

Phase 2: K-Means Clustering Within Canopies
1. Mapper Function:
   - Load Canopies: Each mapper reads the canopies created in the first phase.
   - Emit Data Points: The mapper outputs data points along with their canopy ID for further processing.

2. Reducer Function:
   - Load Data Points: Each reducer receives data points associated with a specific canopy ID.
   - Initialize K-Means: Initialize the K-Means algorithm with a predefined number of clusters (k).
   - Iterate K-Means: Perform K-Means clustering within the canopy using the

expensive distance metric (e.g., Euclidean distance).

- Output Clusters: The reducer outputs the final clusters for each canopy.