**Assignment 3 ( DSCD )**

- Anshak Goel (2020283)

- Deeptorshi Mondal (2020294)

- Sahil Goyal (2020326)

**Overview:**

This implementation utilizes the MapReduce paradigm to perform the K-means clustering algorithm. K-means clustering is an iterative algorithm that partitions a dataset into K distinct clusters, where each data point belongs to the cluster with the nearest mean.

We have implemented the following methods in the Master.py file to incorporate the master functionality:

**getInputData():**

Reads the input data points from the file points3.txt and populates the points list.

**initializeCentroids(points, num\_of\_centroids)**:

Initializes the centroids by randomly selecting num\_of\_centroids data points from the input data.

**callMapper(start\_index, end\_index, iteration\_num, mapper\_id, reducers):**

Invokes the mapper function via gRPC communication, passing the necessary data including the start and end indices of the data points to be processed, iteration number, mapper ID, and current centroids.

**callReducer(iteration\_num, reducer\_id, work\_reducer, mappers):**

Invokes the reducer function via gRPC communication, passing the iteration number, reducer ID, current mapper work assignment, and total number of mappers.

**call\_parallel\_mapper(start\_index, end\_index, iter, i, mappers, reducers):**

Facilitates parallel execution of mapper tasks by assigning tasks to mapper nodes and handling failures.

**call\_parallel\_reducer(iter, i, reducers, mappers):**

Facilitates parallel execution of reducer tasks by assigning tasks to reducer nodes and handling failures.

**main():**

The entry point of the script.

Reads user input for the number of mappers, reducers, centroids, and maximum iterations.

Initializes centroids and coordinates the execution of mapper and reducer tasks in iterations until convergence or maximum iterations are reached.

The short description of the methods implemented in the mapper is the following:

**getInputData():**

Reads the input data points from the file points3.txt and populates the points list.

**MasterAndMapper.mapRequest(self, request, context):**

Receives a mapping request from the master node.

Computes the closest centroid for each data point within the specified range.

Partitions the data points based on their closest centroids and writes them into intermediate files.

**MasterAndMapper.partition(self, pairs, reducers, mapper\_id, iteration):**

Partitions the data points into files corresponding to each reducer.

Creates or updates files (R1.txt, R2.txt, ..., Rn.txt) in the Mappers/M<Mapper\_ID> folder.

**MasterAndMapper.distance(self, point1, point2):**

Computes the Euclidean distance between two points.

**MapperAndReducer.getPairs(self, request, context):**

Retrieves pairs of data points and their closest centroids from the intermediate files generated by mappers.

Sends the pairs to the reducer for further processing.

**main():**

Reads input data.

Configures the gRPC server to listen for incoming requests from the master node.

Registers MasterAndMapper and MapperAndReducer services with the server.

Starts the gRPC server and waits for incoming requests.

The short description of the methods implemented in the Reducer is the following:

**MasterAndReducer.reduceRequest(self, request, context):**

Receives a reduce request from the master node containing the iteration number, reducer ID, work reducer ID, and number of mappers.

Retrieves pairs of data points and their closest centroids from the mappers.

Performs the shuffle and sort phase by gathering pairs from all mappers.

Invokes the reduce method to compute new centroids.

Returns the new centroids to the master node.

**MasterAndReducer.shuffle\_and\_sort(self, reducer\_id, mappers, work\_reducer\_id):**

Connects to each mapper node and retrieves pairs of data points and their closest centroids.

Aggregates the pairs from all mappers for the specified work reducer.

**MasterAndReducer.reduce(self, pairs):**

Computes new centroids by averaging the data points associated with each centroid.

**main():**

Configures the gRPC server to listen for incoming requests from the master node.

Registers the MasterAndReducer service with the server.

Starts the gRPC server and waits for incoming requests.