PROBLEM STATEMENT: Design a distributed application using MapReduce which processes Movie dataset. Recommend the Movie based on the user ratings. Use Movie dataset and process it using a pseudo distribution mode on Hadoop platform.

Step 1: Create a Java Project in Eclipse

1. **Create a New Java Project**:
   - Open Eclipse and go to **File** → **New** → **Java Project**.
   - Name the project (e.g., `Movie`).
   - Click **Finish**.
2. **Create a Package**:
   - In the **Project Explorer**, right-click on `src` → **New** → **Package**.
   - Name the package (e.g., `Movie`).
3. **Create the Classes**:
   - Right-click on the package you created → **New** → **Class**.
   - Create the following classes:
     - **MovieMapper**
     - **MovieReducer**
     - **MovieDriver** (Driver class)

Step 2: Add the Code

Add the following code in their respective classes:

//MovieMapper.java

```
import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;


public class MovieMapper extends MapReduceBase implements Mapper<LongWritable, Text,
Text, Text> {


   public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, Reporter
reporter) throws IOException {

      String[] parts = value.toString().split(" ");

      if (parts.length == 3) {

         String movieId = parts[0];

         String rating = parts[2];

         output.collect(new Text(movieId), new Text(rating));

      }

   }

}
```

//MovieReducer.java

```
import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reporter;


public class MovieReducer extends MapReduceBase implements Reducer<Text, Text, Text,
Text> {
    public void reduce(Text key, Iterator<Text> values, OutputCollector<Text, Text> output,
Reporter reporter) throws IOException {

        int sumRatings = 0;

        int count = 0;


        while (values.hasNext()) {

            String ratingStr = values.next().toString();

            int rating = Integer.parseInt(ratingStr);

            sumRatings += rating;

            count++;

        }


        float avgRating = (float) sumRatings / count;


        output.collect(key, new Text("AverageRating=" + avgRating));

    }

}
```

//MovieDriver.java

```
import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;


public class MovieDriver {
    public static void main(String[] args) throws Exception {
        JobConf conf = new JobConf(MovieDriver.class);
        conf.setJobName("Movie Recommendation based on Ratings");

        conf.setMapperClass(MovieMapper.class);
        conf.setReducerClass(MovieReducer.class);

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(Text.class);

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

        JobClient.runJob(conf);
    }
}
```

Step 3: Build the Project

1. **Build the Project**:
    - Click on **Project** → **Build Project** in Eclipse.
    - Make sure the project compiles without any errors.
2. **Check Build Path**:
    - Go to **Project Explorer** → Right-click on your project → **Build Path** → **Configure Build Path**.
    - Ensure that all the Hadoop JAR files you added are present in the **Libraries** section.
        i. hadoop-common (e.g., `hadoop-common.jar`)

    ii. Hadoop-mapreduce-client-core (e.g., `hadoop-mapreduce-client-core-2.x.x.jar`)

Step 4: Create the JAR File

1. **Export to JAR**:
   - Go to **File** → **Export**.
   - Choose **Java** → **Runnable JAR file**.
   - Choose **Launch configuration** as `Movie`.
   - Select the destination path and name your JAR file (e.g., `Movie.jar`).
   - Click **Finish**.

Step 5: Prepare the Input Files

1. **Create the Input File (`moviedata.txt`)**:
   - The **`moviedata`**`.txt` should be placed in the HDFS directory. Here's the content of the file:

     track1 user1 listen

movie1 user1 5

movie1 user2 4

movie2 user1 2

movie2 user3 5

movie3 user2 3

movie3 user4 4

Upload the Input File to HDFS:

- Use the Hadoop shell to upload the input file to HDFS:

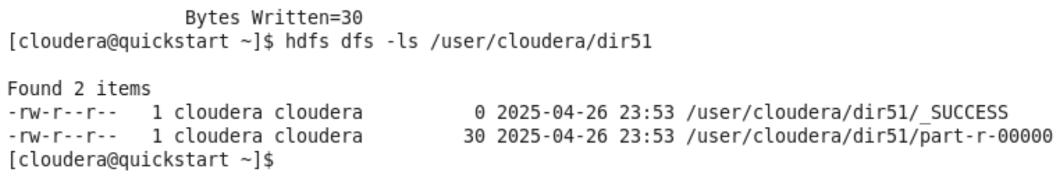
hdfs dfs -put /path/to/moviedata.txt /user/cloudera/moviedata.txt

Step 6: Configure the Run Configuration

1. **Set up Run Configuration**:
    - In Eclipse, go to **Run** → **Run Configurations**.
    - Select **Java Application** and click **New**.
    - In the **Main** tab, select the **Project** (your Hadoop project) and the **Main Class** (`W`).

Step 8: Run the Job

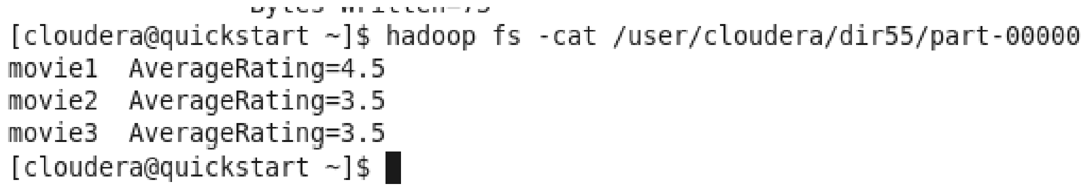
1. **Run the Hadoop Job**:
   [cloudera@quickstart ~]$ hadoop jar /home/cloudera/movie.jar Movie.Movie /user/cloudera/moviedata.txt /user/cloudera/dir51
2. hdfs dfs -ls /user/cloudera/dir51



```
                Bytes Written=30
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera/dir51

Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2025-04-26 23:53 /user/cloudera/dir51/_SUCCESS
-rw-r--r--   1 cloudera cloudera         30 2025-04-26 23:53 /user/cloudera/dir51/part-r-00000
[cloudera@quickstart ~]$
```

3. Hadoop fs -cat /user/cloudera/dir51/part-00000



```
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/dir55/part-00000
movie1  AverageRating=4.5
movie2  AverageRating=3.5
movie3  AverageRating=3.5
[cloudera@quickstart ~]$
```