ASSIGNMENT 5

PROBLEM STATEMENT:

To develop a MapReduce program to analyse Titanic ship data, focusing on calculating the average age of deceased males and determining the count of female casualties in each class.

OBJECTIVE:

• Develop a MapReduce program for analysing Titanic ship data.

• Compute the average age of males who died in the tragedy.

• Determine the number of female casualties in each class.

• Understand and implement the key components of MapReduce: Mapper and Reducer functions.

Hardware Requirements

• A Hadoop cluster for distributed computing.

• Sufficient computational resources on the cluster to handle the input file.

Software Requirements

• Hadoop Installation: Ensure that Hadoop is installed and configured on the cluster.

• Java Development Kit (JDK): MapReduce programs are typically written in Java, so JDK is required for development.

• Text Editor or Integrated Development Environment (IDE): Use a text editor or IDE to write and edit the MapReduce program.

• Titanic Ship Data: The dataset containing information about passengers on the Titanic.

Theory

• Hadoop, a pioneering open-source framework, serves as the bedrock for processing vast datasets in parallel across expansive clusters of commodity hardware. It derives inspiration from Google's infrastructure and is widely utilized, especially in the realm of big data applications.

• In the context of MapReduce, a pivotal paradigm within Hadoop, the workflow involves disassembling colossal datasets into manageable fragments. This segmentation facilitates independent processing through map tasks, fostering efficient parallel computation. Subsequently, a reduction phase orchestrates the synthesis of these processed chunks, employing techniques such as speculative execution for continuous progress and checkpointing for restarting from intermediate stages in case of

failures. In the Map phase, the input data is divided into smaller chunks and processed independently in parallel across multiple nodes in a distributed computing environment. Each chunk is transformed or "mapped" into key-value pairs by applying a user-defined function. The output of the Map phase is a set of intermediate key-value pairs.

• The Reduce phase follows the Map phase. It gathers the intermediate key-value pairs generated by the Map tasks, performs data shuffling to group together pairs with the same key, and then applies a user-defined reduction function to aggregate and process the data. The output of the Reduce phase is the final result of the computation.

• The Map phase, a crucial precursor, acts as the engine for data transformation. Input data undergoes a metamorphosis into key-value pairs, steered by user-defined mapping functions (Fmap). This phase yields intermediate key-value pairs, laying the groundwork for the impending reduction.

• Contrastingly, the Reduction phase assumes the role of data aggregation and synthesis. It assimilates intermediate key-value pairs generated by map tasks, employing techniques such as in-memory merging (Mr) for efficient data aggregation. The orchestrated process involves key grouping through hashing (H) or sorting algorithms, showcasing the intricacies of distributed data processing tasks under the orchestration of Hadoop's job scheduler (JS).

• Input and Output types of a MapReduce job: (input) -> map -> -> combine -> -> reduce -> (output)

• MapReduce allows for efficient processing of large-scale datasets by leveraging parallelism and distributing the workload across a cluster of machines. It simplifies the development of distributed data processing applications by abstracting away the complexities of parallelization, data distribution, and fault tolerance, making it an essential tool for big data processing in the Hadoop ecosystem.

• The Titanic dataset encapsulates a rich array of information about passengers on the ill-fated voyage. Key attributes include passenger class, gender, age, and survival status. This multifaceted dataset serves as a poignant historical record, offering insights into the societal impact of the tragedy. Our MapReduce exploration aims to glean nuanced details from this dataset, shedding light on the human dimension of this historical event.

Code

```
// import libraries import java.io.IOException;
 import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
```

```
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

// Making a class with name Average_age
public class Average_age {
public static class Map extends Mapper<long writable, Text, Text, Int Writable> {
// private text gender variable which
// stores the gender of the person
// who died in the Titanic Disaster private Text gender = new Text();
// private IntWritable variable age will store
// the age of the person for MapReduce. where
// key is gender and value is age
private IntWritable age = new IntWritable();
// overriding map method(run for one time for each record in dataset)
public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException
{
// storing the complete record
// in a variable name line
String line = value.toString();
// splitting the line with ', ' as the
// values are separated with this
// delimiter
String str[] = line.split(", ");

/* checking for the condition where the number of columns in our dataset has to be more than 6. This helps in eliminating the ArrayIndexOutOfBoundsException when the data sometimes is incorrect in our dataset*/

if (str.length > 6)
{
// storing the gender
```

```
// which is in 5th column
gender.set(str[4]);
// checking the 2nd column value in
// our dataset, if the person is
// died then proceed. if ((str[1].equals("0"))) {
// checking for numeric data with
// the regular expression in this column if (str[5].matches("\\d+")) {
// converting the numeric
// data to INT by typecasting int i = Integer.parseInt(str[5]);
// storing the person of age age.set(i);
} } }
// writing key and value to the context
// which will be output of our map phase
context.write(gender, age);
} }
public static class Reduce extends Reducer {
// overriding reduce method(runs each time for every key )
public void reduce(Text key, Iterable values, Context context) throws IOException, InterruptedException
{
// declaring the variable sum which
// will store the sum of ages of people int sum = 0; // Variable l keeps incrementing for // all the value of that key.
 int l = 0;
// foreach loop
for (IntWritable val : values) {
l += 1;
// storing and calculating
// sum of values
sum += val.get();
 } sum = sum / l;
context.write(key, new IntWritable(sum));
} }
```

```
public static void main(String[] args) throws Exception
{
Configuration conf = new Configuration();
@SuppressWarnings("deprecation")
Job job = new Job(conf, "Averageage_survived");
job.setJarByClass(Average_age.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(IntWritable.class);
//job.setNumReduceTasks(0);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setInputFormatClass(TextInputFormat.class); job.setOutputFormatClass(TextOutputFormat.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
Path out = new Path(args[1]);
out.getFileSystem(conf).delete(out);
 job.waitForCompletion(true);
}
 }
```

CONCLUSION:

In this way we have develop a MapReduce program to analyse Titanic ship data, focusing on calculating the average age of deceased males and determining the count of female casualties in each class.

ORAL QUESTION:

1. How do you handle scenarios where there are inconsistencies or errors in the Titanic ship data?
2. How do you handle scenarios where the Titanic ship data is distributed across multiple files or partitions?
3. What approach would you take to determine the count of female casualties in each class?

Code:

```
import pandas as pd
def map_reduce_with_pandas(input_file):
 # Load the dataset
 df = pd.read_csv(input_file)

 # Map: Filter deceased males and transform data for average age
calculation
 deceased_males = df[(df['Survived'] == 0) & (df['Sex'] == 'male')]

 # Reduce: Calculate average age of deceased males
 average_age_deceased_males = deceased_males['Age'].mean()

 # Map: Filter deceased females and transform data for count by class
 deceased_females_by_class = df[(df['Survived'] == 0) & (df['Sex'] ==
'female')]
 # Reduce: Count deceased females by class
 count_deceased_females_by_class =
deceased_females_by_class['Pclass'].value_counts()
 return average_age_deceased_males, count_deceased_females_by_class

# Example usage
input_file = r'D:\BE SEM VIII\CL_IV_Code\titanic.csv' # Update this
to the path of your Titanic dataset CSV file
average_age, female_class_count = map_reduce_with_pandas(input_file)
print(f"Average age of males who died: {average_age:.2f}")
print("Number of deceased females in each class:")
print(female_class_count)
```

Output:

```
Average age of males who died: 31.62
Number of deceased females in each class:
Pclass
3    72
2     6
1     3
Name: count, dtype: int64
```