Cairo University
Faculty of Computers and Artificial Intelligence
Information Systems Department

Managing and Modeling Big Data [2023-2024]

Lab3: MapReduce Examples

Example 1: Finding the Average Age of Male and Female Died in Titanic Disaster

Problem Statement: Analyzing the Titanic Disaster dataset, for finding the average age of male and female persons died in this disaster with MapReduce Hadoop.

1. Open Eclipse on Cloudera platform.
2. Create new project name it “TitanicDataAnalysis”.
3. Add all needed libraries by clicking “Add External Jars” button:
    a. File System → Lib → Hadoop (Add all jars)
    b. File System → Lib → Hadoop → Client (Add all jars)
4. Create three new classes in the project:
    a. TitanicMapper:
        i. Description:
        This class is responsible for parsing the input data (in this case, the Titanic dataset in CSV format) and emitting key-value pairs of the form (gender, age) for each passenger who died. The gender is used as the key, and the age is used as the value.
        ii. Code:

```
import java.io.IOException;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class TitanicMapper extends Mapper<Object, Text, Text,
DoubleWritable> {
```

```
    private Text gender = new Text();
    private DoubleWritable age = new DoubleWritable();

    public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {

      // Split the CSV row by comma
      String[] row = value.toString().split(",");

      // Extract gender and age
      String genderValue = row[4];
      String ageValue = row[5];

      // Check if passenger survived
      if (row[1].equals("0")) {
        // Check gender and emit key-value pair
        if (genderValue.equals("female")) {
          gender.set("female");
          if (!ageValue.isEmpty()) {
            age.set(Double.parseDouble(ageValue));
            context.write(gender, age);
          }
        } else if (genderValue.equals("male")) {
          gender.set("male");
          if (!ageValue.isEmpty()) {
            age.set(Double.parseDouble(ageValue));
            context.write(gender, age);
          }
        }
      }
    }
}
```

b. TitanicReducer:

i. Description:

This class receives the key-value pairs emitted by the mapper and calculates the average age for males and females separately using the approach of computing the sum of ages and the count of passengers of each gender. The reducer then emits two key-value pairs, one for each gender, with the key being the gender and the value being the average age.

ii. Code:

```
import java.io.IOException;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.mapreduce.Reducer;

public class TitanicReducer extends Reducer<Text, DoubleWritable, Text,
DoubleWritable> {

   private DoubleWritable result = new DoubleWritable();

   public void reduce(Text key, Iterable<DoubleWritable> values, Context
context)
         throws IOException, InterruptedException {

      double sum = 0;
      int count = 0;
      for (DoubleWritable val : values) {
         sum += val.get();
         count++;
      }

      double average = sum / count;
      result.set(average);
      context.write(key, result);
   }
}
```

c. TitanicAverageAge:
   i. Description:
   This class is responsible for configuring and running the MapReduce job. It sets the input and output paths, as well as the input and output formats, and specifies the mapper and reducer classes to use. Finally, it initiates the job and waits for it to complete before exiting.
   ii. Code:

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class TitanicAverageAge {

   public static void main(String[] args) throws Exception {

      if (args.length != 2) {
```

```
        System.err.println("Usage: TitanicAverageAge <input path> <output path>");
        System.exit(-1);
    }

    Job job = Job.getInstance();
    job.setJarByClass(TitanicAverageAge.class);
    job.setJobName("Titanic Average Age");

    // Set input and output paths
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    // Set mapper, reducer and output classes
    job.setMapperClass(TitanicMapper.class);
    job.setReducerClass(TitanicReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(DoubleWritable.class);

    // Wait for job to complete
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

5. Export the jar file (Export → Java → Jar → name: "AvgHeightAge.jar") and save it on Cloudera home.
6. Open Terminal:
    a. Copy the data file from local to HDFS:

    ```
    hadoop fs -copyFromLocal titanic_data_new.csv
    ```

    b. Run the application:

    ```
    hadoop jar /home/cloudera/Titanic_Data_Analysis.jar
    TitanicAverageAge titanic_data_new.csv titanicoutput
    ```

    c. Check the output on terminal:

    ```
    hadoop fs -cat titanicoutput/part-r-00000
    ```

    d. Copy the output file to Cloudera's local file system:

    ```
    hadoop fs -copyToLocal titanicoutput/part-r-00000
    ```

Example 2: Sum of even and odd numbers in MapReduce

Problem Statement: Counting the number of even and odd and finding their sum.

1. Open Eclipse on Cloudera platform.
2. Create new project name it “EvenOdd”.
3. Add all needed libraries by clicking “Add External Jars” button:
    a. File System → Lib → Hadoop (Add all jars)
    b. File System → Lib → Hadoop → Client (Add all jars)
4. Create three new classes in the project:
    a. EOMapper:
        i. Description:
        This is the Mapper class that reads the input text file and emits key-value pairs where the key is "even" or "odd" depending on whether the input number is even or odd, and the value is the input number itself.
        ii. Code:

```
// Importing libraries
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
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 EOMapper extends MapReduceBase implements
Mapper<LongWritable,

                        Text, Text, IntWritable> {

    @Override
    // Map function
    public void map(LongWritable key, Text value,
OutputCollector<Text,

    IntWritable> output, Reporter rep)

    throws IOException
```

```
        {
                // Splitting the line into spaces
                String data[] = value.toString().split(" ");

                for (String num : data)
                {

                        int number = Integer.parseInt(num);

                        if (number % 2 == 1)
                        {
                                // For Odd Numbers
                                output.collect(new Text("ODD"), new
IntWritable(number));
                        }

                        else
                        {
                                // For Even Numbers
                                output.collect(new Text("EVEN"),
                                        new IntWritable(number));
                        }
                }
        }
}
```

b. EOReducer:

i. Description:

This is the Reducer class that receives the key-value pairs emitted by the Mapper and computes the sum and count of even numbers and odd numbers separately. It emits two key-value pairs, one with the key "even" and the sum of even numbers as the value, and another with the key "odd" and the sum of odd numbers as the value. The same is done for the count.

ii. Code:

```
// Importing libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
```

```
public class EOReducer extends MapReduceBase implements Reducer<Text,
                                                                IntWritable,
Text, IntWritable> {

        @Override
        // Reduce Function
        public void reduce(Text key, Iterator<IntWritable> value,
        OutputCollector<Text, IntWritable> output, Reporter rep)

        throws IOException
        {

                // For finding sum and count of even and odd
                // you don't have to take different variables
                int sum = 0, count = 0;
                if (key.equals("ODD"))
                {
                        while (value.hasNext())
                        {
                                IntWritable i = value.next();

                                // Finding sum and count of ODD Numbers
                                sum += i.get();
                                count++;
                        }
                }

                else
                {
                        while (value.hasNext())
                        {
                                IntWritable i = value.next();

                                // Finding sum and count of EVEN
Numbers
                                sum += i.get();
                                count++;
                        }
                }

                // First sum then count is printed
                output.collect(key, new IntWritable(sum));
```

```
                output.collect(key, new IntWritable(count));
        }
}
```

c. EODriver:

i. Description:

This is the main Driver class that sets up the MapReduce job by configuring input and output paths, setting the Mapper and Reducer classes, and running the job on the Hadoop cluster.

ii. Code:

```
// Importing libraries
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class EODriver extends Configured implements Tool {

        @Override
        public int run(String[] args) throws Exception
        {
                if (args.length < 2)
                {
                        System.out.println("Please enter valid arguments");
                        return -1;
                }

                JobConf conf = new JobConf(EODriver.class);
                FileInputFormat.setInputPaths(conf, new Path(args[0]));
                FileOutputFormat.setOutputPath(conf, new Path(args[1]));
                conf.setMapperClass(EOMapper.class);
                conf.setReducerClass(EOReducer.class);
                conf.setMapOutputKeyClass(Text.class);
                conf.setMapOutputValueClass(IntWritable.class);
                conf.setOutputKeyClass(Text.class);
                conf.setOutputValueClass(IntWritable.class);

                JobClient.runJob(conf);
```

```
                return 0;
        }

        // Main Method
        public static void main(String args[]) throws Exception
        {
                int exitcode = ToolRunner.run(new EODriver(), args);
                System.out.println(exitcode);
        }
}
```

5. Export the jar file (Export → Java → Jar → name: "EvenOdd.jar") and save it on Cloudera home.
6. Open Terminal:
    a. Copy the data file from local to HDFS:

    ```
    hadoop fs -copyFromLocal EOFile.txt
    ```

    b. Run the application:

    ```
    hadoop jar /home/cloudera/EvenOdd.jar EODriver EOFile.txt EOoutput
    ```

    c. Check the output on terminal:

    ```
    hadoop fs -cat EOoutput/part-00000
    ```

    d. Copy the output file to Cloudera's local file system:

    ```
    hadoop fs -copyToLocal EOoutput/part-r-00000
    ```