Practical - 4

2CS702 – Big Data Analytics

Harshit Gajipara
19BCE059

Aim:

Design MapReduce algorithms to take a very large file of integers and produce as output:

a) The largest integer

b) The average of all the integers.

Steps:

- Create a new project using Ant in apache netbeans
- Add libraries from below path to library and set jdk as 1.8.0

"D:/BDA/hadoop-3.2.1/share/hadoop/common/hadoop-common-3.2.1.jar"

"D:/BDA/hadoop-3.2.1/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.2.1.jar"

- After saving java files, clean and build the project and run hadoop command after uploading num.txt file.

Hadoop jar "C:\Users\HARSHIT\Documents\NetBeansProjects\practical4\dist\practical4.jar" /prac4/num.txt /prac4/output

- Now run this command to view the output

Hdfs dfs –cat /prac4/output/part-r-00000

Code:

`Practical4.java`

```
/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
```

```
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java
to edit this template
 */
package practical4;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
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;

/**
 *
 * @author Harshit
 */
public class Practical4 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws
IOException, InterruptedException, ClassNotFoundException {
        if (args.length != 2) {
            System.err.println("Please specify the input and
output path");
            System.exit(-1);
        }
        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "Practical4");
        job.setJarByClass(Practical4.class);
        job.setMapperClass(map.class);
```

```
        job.setCombinerClass(reduce.class);
        job.setReducerClass(reduce.class);


        job.setMapOutputValueClass(Text.class);
        job.setMapOutputKeyClass(Text.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(DoubleWritable.class);



        FileInputFormat.addInputPath(job, new
Path(args[0]));
        FileOutputFormat.setOutputPath(job, new
Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }

}
```

map.java

```
/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package practical4;

import java.io.IOException;
import static java.lang.Integer.max;
import jdk.nashorn.internal.runtime.JSType;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
```

```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

/**
 *
 * @author Harshit
 */
public class map extends Mapper<LongWritable, Text, Text,
Text> {
    @Override
    protected void map(LongWritable key, Text value, Context
context) throws IOException, InterruptedException {

        String data[] = value.toString().split(" ");
        Double total = 0.0;
        int no = 0;
        int m = Integer.MIN_VALUE;

        for (String x : data) {
            int num = 0;
            if (x.charAt(0) == '-')
                num = -1 * Integer.parseInt(x.substring(1));
            else
                num = Integer.parseInt(x);

            no++;
            total += num;
            m = max(num, m);

        }

        String maxV = Integer.toString(m);
        String tot_freq = Double.toString(total) + "," +
Integer.toString(no);
        context.write(new Text("Max"), new Text(maxV));
        context.write(new Text("tot_freq"), new
Text(tot_freq));
```

```
    }
}
```

reduce.java

```
/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package practical4;

import java.io.IOException;
import static java.lang.Integer.max;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

/**
 *
 * @author Harshit
 */
public class reduce extends Reducer<Text, Text, Text, Text>
{

    @Override
    protected void reduce(Text key, Iterable<Text> values,
Context context) throws IOException, InterruptedException {
        int maxValue = Integer.MIN_VALUE;

        double sum = 0;
        int no = 0;

        System.out.println("Key : " + key.toString());
```

```
        if (key.toString().equals("Max")) {
            for (Text val : values) {
                maxValue = max(maxValue,
Integer.parseInt(val.toString()));
            }
            context.write(new Text("Max"), new
Text(Integer.toString(maxValue)));
        }
        if (key.toString().equals("tot_freq")) {
            for (Text val : values) {
                String[] tokens = val.toString().split(",");
                sum += Double.parseDouble(tokens[0]);
                no += Integer.parseInt(tokens[1]);
            }
            context.write(new Text("Average"), new
Text(Double.toString(sum / no)));
        }
        if (key.toString().equals("Average")) {
            for (Text val : values) {
                context.write(new Text("Average"), new
Text(val.toString()));
            }
        }
    }
}
```

Output:

```
C:\Users\HARSHIT>hdfs dfs -put "C:\Users\HARSHIT\num.txt" /prac4
2022-10-14 12:45:33,341 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remot
eHostTrusted = false

C:\Users\HARSHIT>hadoop jar C:\Users\HARSHIT\Documents\NetBeansProjects\practical4\dist\practical4.jar /prac4/num.txt
/prac4/output
2022-10-14 12:45:51,706 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2022-10-14 12:45:51,931 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2022-10-14 12:45:51,932 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2022-10-14 12:45:52,520 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implemen
t the Tool interface and execute your application with ToolRunner to remedy this.
2022-10-14 12:45:52,619 INFO input.FileInputFormat: Total input files to process : 1
2022-10-14 12:45:52,788 INFO mapreduce.JobSubmitter: number of splits:1
2022-10-14 12:45:52,998 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local2073760362_0001
2022-10-14 12:45:52,998 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-10-14 12:45:53,160 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2022-10-14 12:45:53,162 INFO mapreduce.Job: Running job: job_local2073760362_0001
2022-10-14 12:45:53,163 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2022-10-14 12:45:53,169 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-10-14 12:45:53,173 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under out
put directory:false, ignore cleanup failures: false
```

```
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=14
                Total committed heap usage (bytes)=574095360
        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=1959590
        File Output Format Counters
                Bytes Written=27

C:\Users\HARSHIT>hdfs dfs -cat /prac4/output/part-r-00000
2022-10-14 12:46:07,594 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remot
eHostTrusted = false
Max     198700
Average 50000.0
```

Conclusion:

In this practical, we learnt that using hadoop mapper and reducer class, we can process large quantity of data parallely and faster than normal processing.