MARRI LAXMAN REDDY GROUP OF INSTITUTIONS MLR INSTITUTE OF TECHNOLOGY (UGC AUTONOMOUS) Affiliated to JNTUH, Approved by AICTE Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India NBA NATIONAL BOARD ACCREDITATION NAAC

BIG DATA ANALYTICS LAB

WEEK-4

NAME : M.SRIKAR **BRANCH & SEC :** CSD-B

ROLLNO : 22R21A67A7 **DATE :** 27-09-24

PROBLEM STATEMENT:

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

1) The largest interger

2)The average of all integers

3)The same set of integers ,but with each integer appearing only once

4)The count of the number of distinct integers in the input

Source code:

```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;
import java.util.HashSet;
import java.util.Set;

public class UniqueStatistics {

    public static class StatsMapper extends Mapper<Object, Text, Text, IntWritable> {
        private final static Text maxKey = new Text("max");
        private final static Text sumKey = new Text("sum");
        private final static Text countKey = new Text("count");
        private final static Text distinctKey = new Text("distinct");
        private final static IntWritable one = new IntWritable(1);

        @Override
        public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
            // Parse the integer from the input
            int number = Integer.parseInt(value.toString());


            // Parse the integer from the input
            int number = Integer.parseInt(value.toString());

            // Emit for max value
            context.write(maxKey, new IntWritable(number));

            // Emit for sum and count
            context.write(sumKey, new IntWritable(number));
            context.write(countKey, one);

            // Emit for distinct integers
            context.write(distinctKey, new IntWritable(number));
        }
    }

    public static class StatsReducer extends Reducer<Text, IntWritable, Text, Text> {
        private int max = Integer.MIN_VALUE;
        private int sum = 0;
        private int count = 0;
        private Set<Integer> distinctIntegers = new HashSet<>();

        @Override
        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
        IOException, InterruptedException {
            if (key.equals(new Text("max"))) {
                // Calculate max
                for (IntWritable val : values) {
                    max = Math.max(max, val.get());
                }
            } else if (key.equals(new Text("sum"))) {
                // Calculate sum
                for (IntWritable val : values) {
                    sum += val.get();
                }
            } else if (key.equals(new Text("count"))) {
                // Calculate count
```

```
                }
            } else if (key.equals(new Text("count"))) {
                // Calculate count
                for (IntWritable val : values) {
                    count += val.get();
                }
            } else if (key.equals(new Text("distinct"))) {
                // Collect distinct integers
                for (IntWritable val : values) {
                    distinctIntegers.add(val.get());
                }
            }
        }
        @Override
        protected void cleanup(Context context) throws IOException, InterruptedException {
            // Output results
            context.write(new Text("Largest Integer"), new Text(String.valueOf(max)));
            context.write(new Text("Average"), new Text(String.valueOf(sum / (double) count)));
            context.write(new Text("Distinct Integers"), new Text(distinctIntegers.toString()));
            context.write(new Text("Count of Distinct Integers"), new Text(String.valueOf(distinctIntegers.size())));
        } }
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "unique statistics");
        job.setJarByClass(UniqueStatistics.class);
        job.setMapperClass(StatsMapper.class);
        job.setReducerClass(StatsReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }}
```

```
[cloudera@quickstart Desktop]$ ls
15           22R21A67J7       Enterprise.desktop  Kerberos.desktop  venu
22r21a6763   Desktop          Express.desktop     Parcels.desktop
22R21A67H8\  Eclipse.desktop  fruits              untitled folder
[cloudera@quickstart Desktop]$ cat > /home/cloudera/numbers.txt
1
2
3
4
5
5
3
6
7
8^Z
[1]+  Stopped                 cat > /home/cloudera/numbers.txt
[cloudera@quickstart Desktop]$ cat numbers.txt
cat: numbers.txt: No such file or directory
[cloudera@quickstart Desktop]$ cat /home/cloudera/numbers.txt
1
2
3
4
5
5
3
6
7
[cloudera@quickstart Desktop]$ hdfs dfs -mkdir /inputfolder3
[cloudera@quickstart Desktop]$ hdfs dfs -put /home/cloudera/numbers.txt /inputfo
lder3
[cloudera@quickstart Desktop]$ hdfs dfs -cat /inputfolder3/numbers.txt
1
2
3
4
5
5
3
6
```

```
[cloudera@quickstart Desktop]$ hadoop jar /home/cloudera/UniqueStatistics.jar UniqueStatistics /inputfolder3 /outdir
24/09/27 01:25:19 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
24/09/27 01:25:20 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and
execute your application with ToolRunner to remedy this.
24/09/27 01:25:20 INFO input.FileInputFormat: Total input paths to process : 1
24/09/27 01:25:20 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)
24/09/27 01:25:20 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
24/09/27 01:25:40 INFO mapreduce.Job:  map 100% reduce 100%
24/09/27 01:25:40 INFO mapreduce.Job: Job job_1727424510013_0001 completed successfully
24/09/27 01:25:40 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=429
                FILE: Number of bytes written=287211
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=139
                HDFS: Number of bytes written=99
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=3781
                Total time spent by all reduces in occupied slots (ms)=3155
                Total time spent by all map tasks (ms)=3781
                Total time spent by all reduce tasks (ms)=3155
                Total vcore-milliseconds taken by all map tasks=3781
                Total vcore-milliseconds taken by all reduce tasks=3155
                Total megabyte-milliseconds taken by all map tasks=3871744
                Total megabyte-milliseconds taken by all reduce tasks=3230720
        Map-Reduce Framework
                Map input records=9
                Map output records=36
                Map output bytes=351
                Map output materialized bytes=429
                Input split bytes=121
                Combine input records=0
                Combine output records=0
                Reduce input groups=4
                Reduce shuffle bytes=429
                Reduce input records=36
                Reduce output records=4
```

```
                Total committed heap usage (bytes)=226365440
        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=18
        File Output Format Counters
                Bytes Written=99
[cloudera@quickstart Desktop]$ hdfs dfs -cat /outdir/part-r-00000
Largest Integer 7
Average 4.0
Distinct Integers       [1, 2, 3, 4, 5, 6, 7]
Count of Distinct Integers      7
[cloudera@quickstart Desktop]$
```