

Assignment 2 – Output

Code: Word Count

```
import java.io.IOException;
import java.util.StringTokenizer;

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;

public class WordCount {

    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context
            ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
}
```

```

public static class IntSumReducer
    extends
    Reducer<Text,IntWritable,Text,IntWritable> { private
    IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable>
        values, Context context
        ) throws IOException,
        InterruptedException { int sum = 0;
        for (IntWritable val :
            values) { sum +=
                val.get();
            }
        result.set(sum);
        context.write(key,
            result);
        }
    }
}

```

```

public static void main(String[] args) throws
    Exception { Configuration conf = new
    Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.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:~$ ls
cloudera-manager  Desktop  Downloads  enterprise-deployment.json  lib  Music  Processfile.txt  Templates  workspace
ls -l
cloudera-manager Desktop Downloads enterprise-deployment.json lib Music Processfile.txt Templates WordCount.jar
ls -l
cloudera@quickstart:~$ pwd
~/cloudera
cloudera@quickstart:~$ cat > /home/cloudera/Processfile.txt
vaishnavi khose
vaishnavi khose
vaishnavi khose
prajakta
prajakta
cloudera
cloudera
2
^C^C Stopped
cloudera@quickstart:~$ cat /home/cloudera/Processfile.txt
vaishnavi khose
vaishnavi khose
vaishnavi khose
prajakta
prajakta
cloudera
cloudera
cloudera
cloudera
cloudera
cloudera
cloudera
cloudera
cloudera@quickstart:~$ hdfs dfs -ls /
Found 6 items
drwxr-xr-x - hbase supergroup 0 2022-01-20 09:45 /hbase
drwxr-xr-x - cloudera supergroup 0 2022-01-20 01:18 /inputfile1
drwxr-xr-x - solr solr 0 2015-06-09 03:38 /solr
drwxr-xr-x - hdfs supergroup 0 2022-01-09 07:13 /tmp
drwxr-xr-x - hdfs supergroup 0 2015-06-09 03:30 /user
drwxr-xr-x hdfs supergroup 0 2015-06-09 03:36 /var
cloudera@quickstart:~$ hdfs dfs -mkdir /inputfolder1
cloudera@quickstart:~$ hdfs dfs -put /home/cloudera/Processfile.txt /inputfolder1/
cloudera@quickstart:~$ hdfs dfs -cat /inputfolder1/Processfile.txt
vaishnavi khose
vaishnavi khose

cloudera@quickstart:~$ hadoop jar /home/cloudera/WordCount.jar WordCount /inputfolder1/Processfile.txt /out1
22/01/20 02:01:10 INFO client.RMProxy: Connecting to ResourceManager at /s.s.s.s:8032
22/01/20 02:01:10 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and ex
ecute your application with ToolRunner to remedy this.
22/01/20 02:01:11 INFO input.FileInputFormat: Total input paths to process : 1
22/01/20 02:01:11 INFO mapreduce.JobSubmitter: number of splits:1
22/01/20 02:01:11 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1642668217844_0001
22/01/20 02:01:11 INFO YarnClientImpl: Submitted application application_1642668217844_0001
22/01/20 02:01:12 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1642668217844_0001
22/01/20 02:01:12 INFO mapreduce.Job: Running job: job_1642668217844_0001
22/01/20 02:01:22 INFO mapreduce.Job: Job job_1642668217844_0001 running in user mode : false
22/01/20 02:01:22 INFO mapreduce.Job: map 0% reduce 0%
22/01/20 02:01:28 INFO mapreduce.Job: map 100% reduce 0%
22/01/20 02:01:36 INFO mapreduce.Job: map 100% reduce 100%
22/01/20 02:01:36 INFO mapreduce.Job: Job job_1642668217844_0001 completed successfully
22/01/20 02:01:36 INFO mapreduce.Job: Counters: 49
File System Counters
FILE: Number of bytes read=76
FILE: Number of bytes written=220767
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=227
HDFS: Number of bytes written=52
HDFS: Number of read operations=0
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)=4299
Total time spent by all reduces in occupied slots (ms)=0
Total time spent by all map tasks (ms)=4299
Total time spent by all reduce tasks (ms)=0
Total vcore-seconds taken by all map tasks=4299
Total vcore-seconds taken by all reduce tasks=0
Total megabyte-seconds taken by all map tasks=4402176
Total megabyte-seconds taken by all reduce tasks=5992352
Map-Reduce Framework
Map input records=9
Map output records=12
Map output bytes=149
Map output materialized bytes=78
Input split bytes=28
Combine input records=12
Combine output records=5
Reduce input groups=5
Reduce shuffle bytes=78
Reduce input records=5
Reduce output records=5
Spilled Records=10
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=117
CPU time spent (ms)=1050
Physical memory (bytes) snapshot=34946624
Virtual memory (bytes) snapshot=380732064
Total committed heap usage (bytes)=245378880
Shuffle Errors
BAD ID=0
CORRUPTED=0
IO ERROR=0
WRONG LENGTH=0
WRONG MAP=0
WRONG REDUCE=0
File Input Format Counters
Bytes Read=161
File Output Format Counters
Bytes Written=52
cloudera@quickstart:~$ hdfs dfs -ls /out1
Found 2 items
-rw-r--r-- 1 cloudera supergroup 0 2022-01-20 02:01 /out1/SUCCESS
-rw-r--r-- 1 cloudera supergroup 52 2022-01-20 02:01 /out1/part-r-00000
cloudera@quickstart:~$ hdfs dfs -cat /out1/part-r-00000
akhata 1
cloudera 3
khose 3
prajakta 2
vaishnavi 3
cloudera@quickstart:~$
```

Code: IP COUNT

```
package ass2;

import java.io.*;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

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 org.apache.hadoop.util.GenericOptionsParser;

public class logfile {

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

        Configuration c = new Configuration();

        String[] files = new GenericOptionsParser(c, args).getRemainingArgs();

        Path input = new Path(files[0]);

        Path output = new Path(files[1]);

        Job j = new Job(c, "maxdurationip");

        j.setJarByClass(logfile.class);

        j.setMapperClass(MapForMaxDurationIP.class);

        j.setReducerClass(ReduceForMaxDurationIP.class);

        j.setOutputKeyClass(Text.class);

        j.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(j, input);

        FileOutputFormat.setOutputPath(j, output);

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

    }

    public static class MapForMaxDurationIP extends Mapper<LongWritable, Text, Text,
```

```

IntWritable> {
    public void map(LongWritable key, Text value, Context con) throws IOException,
        InterruptedException {
        String line = value.toString();
        String[] parts = line.split(",");
        String ip = parts[2].trim();
        int duration = Integer.parseInt(parts[4].trim());
        Text outputKey = new Text(ip);
        IntWritable outputValue = new IntWritable(duration);
        con.write(outputKey, outputValue);
    }
}

public static class ReduceForMaxDurationIP extends Reducer<Text, IntWritable, Text,
    IntWritable> {
    private Text maxIp = new Text();
    private IntWritable maxDuration = new IntWritable(Integer.MIN_VALUE);
    private Text minIp = new Text();
    private IntWritable minDuration = new IntWritable(Integer.MAX_VALUE);
    private Text aip = new Text();
    private IntWritable td = new IntWritable(Integer.MIN_VALUE);
    public void reduce(Text ip, Iterable<IntWritable> durations, Context con)
        throws IOException, InterruptedException {
        int totalDuration = 0;
        // int count = 0;
        for (IntWritable duration : durations) {
            totalDuration += duration.get();
            // count++;
        }
        aip.set(ip);
        td.set(totalDuration);
        con.write(aip, td);
        // totalDuration = totalDuration/count;
    }
}

```

```

if (totalDuration > maxDuration.get()) {
maxIp.set(ip);
maxDuration.set(totalDuration);
}
if (totalDuration < minDuration.get()) {
minIp.set(ip);
minDuration.set(totalDuration);
}
}
}
@Override
protected void cleanup(Context con) throws IOException, InterruptedException {
con.write(new Text("\nIp with Max Active Time : " + maxIp.toString()), maxDuration);
con.write(new Text("\nIp with Min Active Time : " + minIp.toString()), minDuration);
}
}
}

```





