Yatish Kelkar TE IT 8001

# Assignment 2 : Word Count

* Aim: Design a distributed application using MapReduce which process a text file. List out the count of each word occuring in the file.

* Theory:

Q1) Explain the need & concept of MapReduce ?

Ans: - MapReduce is a software framework which is used to write applications to process huge amounts of data simultaneously on a large number of nodes in reliable way.

- MapReduce is required when we need to process huge amounts of data that cannot be handled with regular frameworks.

- MapReduce primarily refers to two tasks:

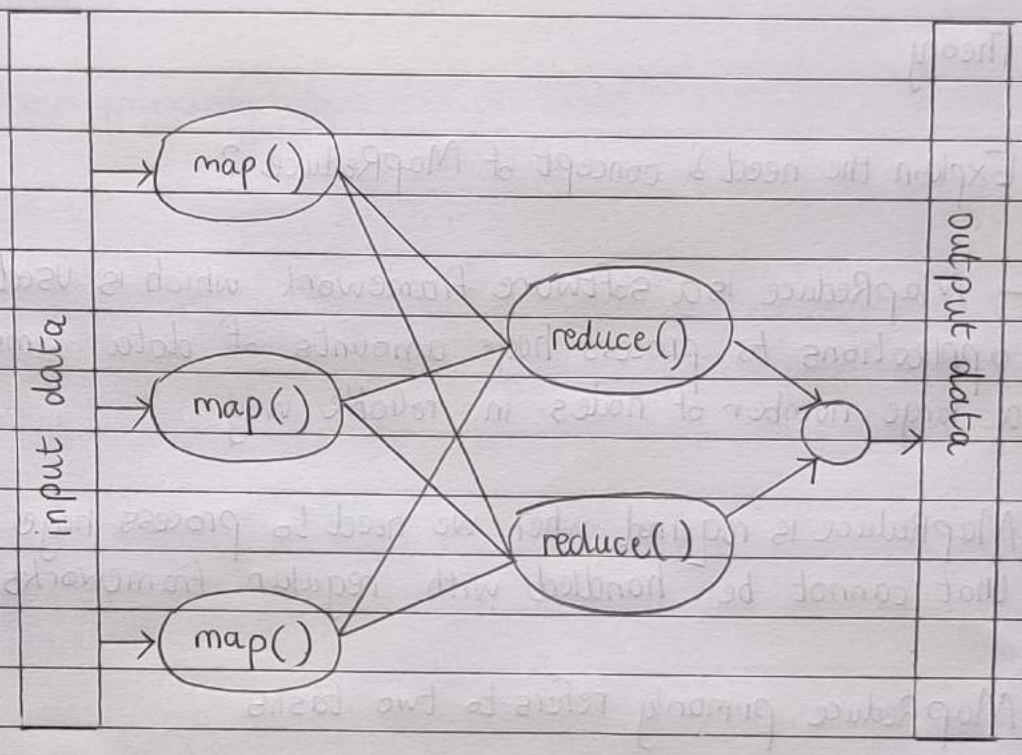
1) Map: Here one set of data is converted into another set of data in which the individual elements are broken down into tuples, ie into key/ value pairs.

2) Reduce: This task takes the output of the map task as its input and combines the data tuples into a smaller tuples. It is

always performed after the map task.

- Advantages of MapReduce framework :

1) Easy to scale
2) Handles task scheduling, monitoring, failure & execution
3) Fault tolerant
4) Simple & easy to understand
5) Has support for unstructed data




Q2) Explain the following :

1) Job Tracker

Ans : - Job Tracker is the master for job management, scheduling & execution in the Hadoop framework.

- Initially the user copies files into HDFS with the -put or the - copyFromLocal commands.

- The job is submitted via the job tracker. It runs on the same node which runs other jobes on data nodes.

- The job is initialized in the job queue & the job tracker creates maps & reduces. The map & reduce tasks will depend on the input programs that user provides.

- Job tracker primarily performs 4 tasks:

i) Resource management
ii) Resource Availability
iii) Monitoring
iv) Scheduling

2) Task Tracker

Ans: - The task tracker accepts tasks assigned by job tracker on the master node while itself running on slave nodes.

- It divides the JVM (Java Virtual Machine) processes & threads to run these tasks. The task tracker reports the progress of these tasks & health status.

- Hadoop maintains 3 lists for task trackers :

i) Blacklist : Used to blacklist a task tracker if performance is not optimal or unstable.

ii) Grey list : a list of potentially problematic nodes

iii) Excluded list : list of excluded task trackers.

* Conclusion : Using the MapReduce framework, application to design a distributed app to process a text file & list out word count has been successfully designed.

## Mapper class : WordMapper.java

```
package Words;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

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

public class WordMapper extends Mapper<LongWritable, Text, Text, IntWritable>{

    public void map(LongWritable key, Text value, Context con) throws
IOException, InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while(tokenizer.hasMoreTokens())
            con.write(new Text(tokenizer.nextToken()), new IntWritable(1));
    }

}
```

## Reducer Class : WordReducer.java

```
package Words;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class WordReducer extends Reducer<Text, IntWritable, Text, IntWritable>
{

    public void reduce(Text word, Iterable<IntWritable> values, Context con)
throws IOException, InterruptedException {

        int sum = 0;

        for(IntWritable value : values) {
            sum += value.get();
        }

        con.write(word, new IntWritable(sum));
    }
}
```

## Driver Class : WordDriver.java

```java
package Words;

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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordDriver {

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

        Job job = new Job(new Configuration(), "WordCount");

        job.setJarByClass(Words.WordDriver.class);
        job.setMapperClass(Words.WordMapper.class);
        job.setReducerClass(Words.WordReducer.class);

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

        FileInputFormat.addInputPath(job, new Path(args[1]));
        FileOutputFormat.setOutputPath(job, new Path(args[2]));

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

## Output Screenshot