|  |  |
| --- | --- |
| EXP NO: 4 | Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm |
|  |
| DATE: |  |

**AIM: -**

**BACKGROUND THEORY: -**

# **PROCEDURE: -**

* Switch to superuser mode using sudo su.
* Create a text file on your local filesystem that contains some sample data (e.g., input.txt).
* Put the file into HDFS (Hadoop Distributed File System)
* The mapper reads the input line by line, splits each line into words, and outputs each word as a key with a value of 1.
* The reducer sums the counts for each word emitted by the mapper and outputs the word along with its total count
* Compile the Java Code
* Run the MapReduce Job
* Once the job is complete, you can check the output by viewing the result file in HDFS

# **CODING: -**

* sudo su
* WordCountMapper.java
  + 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.mapreduce.Mapper;
  + public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
  + private final static IntWritable one = new IntWritable(1);
  + private Text word = new Text();
  + @Override
  + protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
  + String[] words = value.toString().split("\\s+");
  + for (String str : words) {
  + word.set(str.replaceAll("[^a-zA-Z]", "").toLowerCase()); // Normalize word
  + if (!word.toString().isEmpty()) {
  + context.write(word, one);
  + }
  + }
  + }
  + }
* WordCountReducer.java
  + import java.io.IOException;
  + import org.apache.hadoop.io.IntWritable;
  + import org.apache.hadoop.io.Text;
  + import org.apache.hadoop.mapreduce.Reducer;
  + public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
  + @Override
  + protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
  + int sum = 0;
  + for (IntWritable val : values) {
  + sum += val.get();
  + }
  + context.write(key, new IntWritable(sum));
  + }
  + }
* WordCount.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.lib.input.FileInputFormat;
  + import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
  + public class WordCount {
  + 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(WordCountMapper.class);
  + job.setReducerClass(WordCountReducer.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);
  + }
  + }
* Steps to Run the Code
  + hadoop com.sun.tools.javac.Main WordCount.java
  + jar cf wordcount.jar WordCount\*.class
  + hadoop jar wordcount.jar WordCount /input /output
  + hdfs dfs -cat /output/part-r-00000

# **OUTPUT: -**

