

BIG DATA FRAMEWORKS

CSE6001

Prof. Ramesh Ragala

August 19, 2019

- **Problem**

- For a given a set of text documents, the program has to counts the number of occurrences of each word.
- Word Count is a simple and easy to understand algorithm which can be implemented as a Map Reduce application easily.
- The algorithm consists of three main sections
 - Driver Program (Main Program) → Configurations
 - Mapper → Produce key - value pairs form data
 - Reduce → Produce Results in the form of key-value pair

• Writing Mapper Class

- It is created by extending the `org.apache.hadoop.mapreduce.Mapper` Class.
- The `map()` is implemented by overriding the `map` method in the Mapper Class
- The `Mapper()` will take four arguments.
- The input key-value pair and output key-value pair need not be of same type
- Input parameters: Key \rightarrow Byteoffset of the input file(line number).
Value \rightarrow Line of text in the corresponding byteoffset position
- Output Parameters: (word,1) for each word it reads in the line

- **Pseudo Code for Mapper in Hadoop Map Reduce**

```
1: class MAPPER
2:   method MAP(docid  $a$ , doc  $d$ )
3:     for all term  $t \in$  doc  $d$  do
4:       EMIT(term  $t$ , count 1)
```

MAPPER CODE IN JAVA

```
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.*;
import java.util.StringTokenizer;

public class WordCountMapper extends Mapper<LongWritable,Text,Text,IntWritable> {
    private static final IntWritable one = new IntWritable(1);
    private Text word = new Text();

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

• Writing Reducer Class

- It is created by extending the `org.apache.hadoop.mapreduce.Reducer` Class.
- The `reduce()` is implemented by overriding the `reduce` method in the Reducer Class
- It collects all the intermediate key-value pairs generated by the multiple map functions.
- After this, it will sum up all the occurrences of each word and output a key-value pair for each word in the text documents as.

- **Pseudo Code for Reducer in Hadoop Map Reduce**

```
1: class REDUCER
2:   method REDUCE(term  $t$ , counts  $[c_1, c_2, \dots]$ )
3:      $sum \leftarrow 0$ 
4:     for all count  $c \in$  counts  $[c_1, c_2, \dots]$  do
5:        $sum \leftarrow sum + c$ 
6:     EMIT(term  $t$ , count  $sum$ )
```

REDUCER CODE IN JAVA

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;
public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable>
{
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
        IOException, InterruptedException
    {
        int sum = 0;
        for (IntWritable i: values)
        {
            sum = sum + i.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```


• **Writing Driver Program - Steps**

- Job Name : name of this Job
- Executable (Jar) Class: the main executable class.
- Mapper Class: class which overrides the "map" function.
- Reducer: class which override the "reduce" function.
- Output Key: type of output key.
- Output Value: type of output value.
- File Input Path
- File Output Path

DRIVER CODE IN 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 WordCountDriver {
    public static void main(String[] args) throws Exception
    {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "WordCount");
        //Job job = new Job(new Configuration(), "WordCount");
        job.setJarByClass(WordCountDriver.class);
        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.waitForCompletion(true);
        //System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```