Lakshman Madhav Kollipara
ONID-kollipal
CS540 – Database Management Systems

Assignment-3

CS540- Database Management Systems

Arash Termehchy

Source code: WordCount.java, Map.Java, Reduce.Java

Wordcount.java:

```
package kollipal_Wordcount;

import java.util.*;
import java.io.*;
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.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

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

		Configuration conf = new Configuration();
		Job job = new Job(conf,"kollipal_WordCount");
		job.setJarByClass(WordCount.class);
		//Job job =Job.getInstance(conf);
		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(IntWritable.class);
		job.setJarByClass(Map.class);
		job.setMapperClass(Map.class);
		job.setJarByClass(Reduce.class);
		job.setReducerClass(Reduce.class);
		job.setInputFormatClass(TextInputFormat.class);
		job.setOutputFormatClass(TextOutputFormat.class);
		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job, new Path(args[1]));
		job.waitForCompletion(true);
	}


}
```

Lakshman Madhav Kollipara
ONID-kollipal
CS540 – Database Management Systems
Map.Java:

```
package kollipal_Wordcount;

import java.util.*;
import java.io.*;
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.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import java.util.StringTokenizer;

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 Map extends Mapper<LongWritable, Text, Text, IntWritable>
{
	private final static 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 tokenizer = new StringTokenizer(line);
			while (tokenizer.hasMoreTokens())
			{
				word.set(tokenizer.nextToken());
				context.write(word, one);
			}
	}
}
```

Lakshman Madhav Kollipara
ONID-kollipal
CS540 – Database Management Systems
Reduce.java:

```
package kollipal_Wordcount;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.Reducer.Context;
import java.io.*;
import java.util.*;


public class Reduce extends Reducer<Text,IntWritable,Text,IntWritable>
{
	int max_sum = 0;
	Text max_occured_key = new Text();
	public void reduce(Text key, Iterable<IntWritable> values, Context context)
				throws IOException, InterruptedException
				{
					int sum = 0;

					for(IntWritable value: values){
					    sum+= value.get();
					 }
					if (sum > max_sum)
						{
					    max_sum = sum;
					    max_occured_key.set(key);
						}
					//context.write(key, new IntWritable(sum));

				}
					@Override
					protected void cleanup(Context context)
					throws IOException,InterruptedException
					  {
					    context.write(max_occured_key, new
IntWritable(max_sum));
					  }
	}
```

JAR file: kollipal_WordCount.Jar

Hadoop Version: Hadoop 2.6.0

HDP Version: HDP 2.2

Lakshman Madhav Kollipara
ONID-kollipal
CS540 – Database Management Systems

Input:

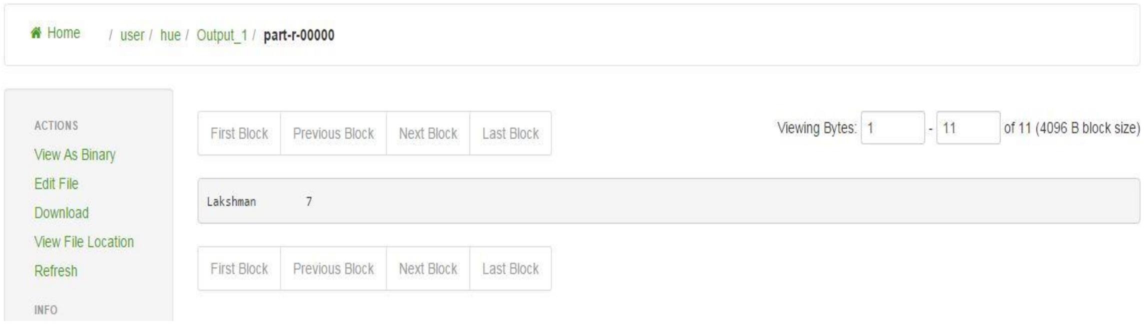
File-1.txt	File-2.txt
Lakshman	Lakshman
Madhav	Madhav
Kollipara	Kollipara
Oregon	Oregon
State	State
University	University
Corvallis	Corvallis
Lakshman	Lakshman
Madhav	Madhav
Kollipara	Kollipara
Oregon	Oregon
State	State
University	University
Corvallis	Corvallis
Lakshman	Lakshman
Lakshman	

Output:

Lakshman	7

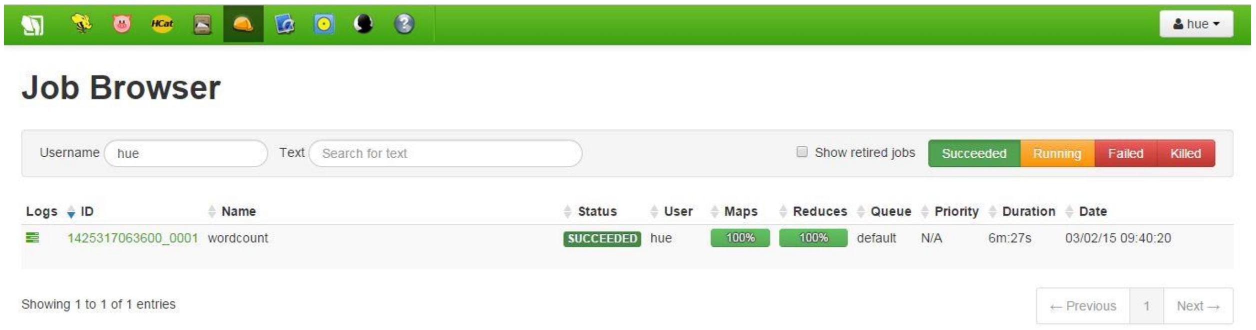
Intermediate Step: (Total occurrence of each word)

Corvallis	4
Kollipara	4
Lakshman	7
Madhav	4
Oregon	4
State	4
University	4



```
Lakshman	7
```

Job Browser: 100% map and 100% reduce Successful



Logs	ID	Name	Status	User	Maps	Reduces	Queue	Priority	Duration	Date
	1425317063600_0001	wordcount	SUCCEEDED	hue	100%	100%	default	N/A	6m:27s	03/02/15 09:40:20

Showing 1 to 1 of 1 entries

← Previous 1 Next →

Description:

Implementing a Hadoop program that takes text files as input and finds the word with maximum frequency in the whole data-set using MapReduce functionality of Hadoop.

Map Reduce has 3 phases: Map, Shuffle and Reduce.

Shuffle is inbuilt, After mapping the Intermediate results are shuffled by Hadoop.

The Mapper of Map phase takes a collection of data and convert it into another set of data, where individual elements are broken into <key,value> pairs.

These <key,value> pairs are shuffled by shuffle function and passed to Reducer.

The Reducer of Reduce phase takes the output from map phase as input and intermediate <key,value> pairs are aggregated into smaller set of <key,value> pairs as an output.

Lakshman Madhav Kollipara
ONID-kollipal
CS540 – Database Management Systems

Map Function:

```
public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException
```

key - input key.
value - input value.
context – write method in context is used to write the output to reducer.

Or Instead of Context, You can use:
output - collects mapped keys and values. OutputCollector<key,value> adds a key/value pair to the output.
reporter - To report progress or just indicate that they are alive

```
String line = value.toString();
StringTokenizer tokenizer = new StringTokenizer(line);
```

Converting value to string and storing it in a line variable and passing as an argument to tokenizer.

```
while (tokenizer.hasMoreTokens())
{
	word.set(tokenizer.nextToken());
	context.write(word, one);
}
```

If more tokens available, then enters while loop. In while loop, we are passing next available token to word, which is a new value object.

`context.write(word, one)` collects <key,value> pair and increments if it encounters that word in the file and sends as output.

For example:

Input: hi hi hi

`context.write(hi, one)` = hi [1,1,1]

Because we set one= IntWritable[1]

Lakshman Madhav Kollipara
ONID-kollipal
CS540 – Database Management Systems

Reduce Function:

```
public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException
```

key - Input key from map.
Iterator – For Iterating the count of one word found
values - input value from map.
context – write method in context is used to write the output.

Or Instead of Context, You can use:
output - collects mapped keys and values. OutputCollector<key,value> adds a key/value pair to the output.
reporter - To report progress or just indicate that they are alive

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

For each value in values, enters for each() loop and increments the sum for each occurrence of word in file. i.e. how many times each word is appeared.

```
if (sum > max_sum)
{
	max_sum = sum;
	max_occured_key.set(key);
}
```

To find the word with highest occurrence, Initializing max_sum to 0 and creating new Text() variable called max_occured_key at the beginning. If the sum is greater than max_sum, then assign that sum to max_sum and set the key value as max_occured_key.So at the end, max_sum will have the sum of highest occurring word in the file and max_occured_key will have the key value which has highest sum i.e. key with highest occurence.

```
protected void cleanup(Context context)
throws IOException,InterruptedException
{
	context.write(max_occured_key, new IntWritable(max_sum));
}
```

`cleanup(Context context)` is used to print single value as output, i.e. Usually reducer prints output of all keys and values. But we are supposed to get only key with highest occurrence as output. So used cleanup.

`context.write(max_occured_key, new IntWritable(max_sum))` prints the key that has high occurrence and its value i.e. max_sum.

Example: Input = hi [1,1,1],

Output = hi [3]