

Summary of the demo given in class (11/19/2015)

Wordcount Program in Hadoop:

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

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.input.TextInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


/**
 *
 * @author Soumyava
 */
public class WordCount {

    //mapper

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

        @Override

        public void map(LongWritable key, Text value, Context c) throws IOException, InterruptedException{

            String str = value.toString();

            String[] strList = str.split(" ");

            //emit each word with 1 frequency
```

```

        for(String s:strList){
            c.write(new Text(s),new Text("1"));
        }
    }
}

//reducer

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

    @Override

    public void reduce(Text key, Iterable<Text>values, Context c) throws IOException,InterruptedException{

        int count = 0;

        for(Text val:values){

            count += 1;

        }

        c.write(key, new Text(""+count));

    }

}

public static void main(String[] args) throws IOException, ClassNotFoundException,
InterruptedException{

    Configuration conf = new Configuration();

    Job j2 = new Job(conf);

    j2.setJobName("Wordcount job");

    j2.setJarByClass(WordCount.class);

    //Mapper input and output

    j2.setMapOutputKeyClass(Text.class);

    j2.setMapOutputValueClass(Text.class);

    //Reducer input and output

    j2.setOutputKeyClass(Text.class);

    j2.setOutputValueClass(Text.class);

    //file input and output of the whole program

```

```
j2.setInputFormatClass(TextInputFormat.class);
j2.setOutputFormatClass(TextOutputFormat.class);

//Set the mapper class
j2.setMapperClass(WordMapper.class);
//set the combiner class for custom combiner
//j2.setCombinerClass(WordReducer.class);
/Set the reducer class
j2.setReducerClass(WordReducer.class);
//set the number of reducer if it is zero means there is no reducer
//j2.setNumReduceTasks(0);
FileOutputFormat.setOutputPath(j2, new Path(args[1]));
FileInputFormat.addInputPath(j2, new Path(args[0]));
j2.waitForCompletion(true);
}
}
```

Custom Partitioner:

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
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 org.apache.hadoop.mapreduce.Job;
```

```

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

/**
 *
 * @author Soumyava
 */
public class CustomPartitioner {

    public static class PartMapper extends Mapper<LongWritable, Text, Text, Text> {

        @Override

        public void map(LongWritable key, Text value, Context c) throws IOException, InterruptedException{

            String[] strList = value.toString().split(",");

            String gender = strList[2];

            String nameAgeScore = strList[0]+","+strList[1];

            c.write(new Text(gender), new Text(nameAgeScore));

        }

    }

    public static class PartPartitioner extends Partitioner<Text, Text>{

        @Override

        public int getPartition(Text key, Text value, int noOfReducers) {

            int age = Integer.parseInt(value.toString().split(",")[1]);

            if(noOfReducers == 0)

                return 0;

            if(age <= 20){

                return 0;

```

```

    }

    else if (age >20 && age <=50){

        return 1%noOfReducers;

    }

    else

        return 2%noOfReducers;

    }

}

```

```

public static class PartReducer extends Reducer<Text,Text,Text,Text>{

    @Override

    public void reduce(Text key, Iterable<Text> values, Context c) throws IOException,
    InterruptedException{

        for(Text val: values){

            c.write(key,val);

        }

    }

}

public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException{

    Configuration conf = new Configuration();

    Job j2 = new Job(conf,"Custom Partitioner");

    j2.setJarByClass(CustomPartitioner.class);


    j2.setMapOutputKeyClass(Text.class);

    j2.setMapOutputValueClass(Text.class);


    j2.setOutputKeyClass(Text.class);

```

```

j2.setOutputValueClass(Text.class);

j2.setInputFormatClass(TextInputFormat.class);
j2.setOutputFormatClass(TextOutputFormat.class);

j2.setMapperClass(PartMapper.class);
j2.setReducerClass(PartReducer.class);
j2.setPartitionerClass(PartPartitioner.class);
j2.setNumReduceTasks(3);
FileOutputFormat.setOutputPath(j2, new Path(args[1]));
FileInputFormat.addInputPath(j2, new Path(args[0]));
j2.waitForCompletion(true);
}
}

```

Compiling the program and making a jar:

To compile the program:

```

javac -Xlint -classpath ${HADOOP_PREFIX}/share/hadoop/common/hadoop-common-
2.2.0.jar:${HADOOP_PREFIX}/share/hadoop/mapreduce/hadoop-mapreduce-client-core-
2.2.0.jar:${HADOOP_PREFIX}/share/hadoop/hdfs/hadoop-hdfs-
2.2.0.jar:${HADOOP_PREFIX}/share/hadoop/common/lib/hadoop-annotations-
2.2.0.jar:${HADOOP_PREFIX}/share/hadoop/common/lib/log4j-
1.2.17.jar:${HADOOP_PREFIX}/share/hadoop/common/lib/commons-cli-1.2.jar -d WordCount_classes
WordCount.java

```

```

${HADOOP_PREFIX}/share/hadoop/common/hadoop-common-
2.2.0.jar:${HADOOP_PREFIX}/share/hadoop/mapreduce/hadoop-mapreduce-client-core-
2.2.0.jar:${HADOOP_PREFIX}/share/hadoop/hdfs/hadoop-hdfs-
2.2.0.jar:${HADOOP_PREFIX}/share/hadoop/common/lib/hadoop-annotations-
2.2.0.jar:${HADOOP_PREFIX}/share/hadoop/common/lib/log4j-
1.2.17.jar:${HADOOP_PREFIX}/share/hadoop/common/lib/commons-cli-1.2.jar -> path to Hadoop
dependencies separated by ":" for adding multiple dependencies. This are the minimum number of
dependencies you need when using Hadoop-2.2.0

```

WordCount_classes -> the folder where all .class files will be dumped

WordCount.java ->the java file

To make the jar:

jar -cvf WordCount.jar -C WordCount_classes .

WordCount.jar -> jar name should be same as the name of the main class.

To execute the jar:

- From the terminal on the Ubuntu.

Use the following command

hadoop jar WordCount.jar "Input path from the dfs" "output path in the dfs"

- To execute on aws.
Use the same jar upload it to s3 and configure the custom jar as mentioned in the AWS Hadoop EMR Guide.docx on page 15.

Few commands to use the hdfs (Hadoop Distributed File System):

To see the file or directory use `hadoop dfs -ls` or `hadoop fs -ls`

To delete the file use `hadoop fs -rm "path to file name"`

To delete the folder use `hadoop fs -rmr "path to folder"`

Likewise for more commands you can use `hadoop fs -help`