Nick Petty

# CAP 6315 Social Networks and Big Data Analytics

# Homework 4

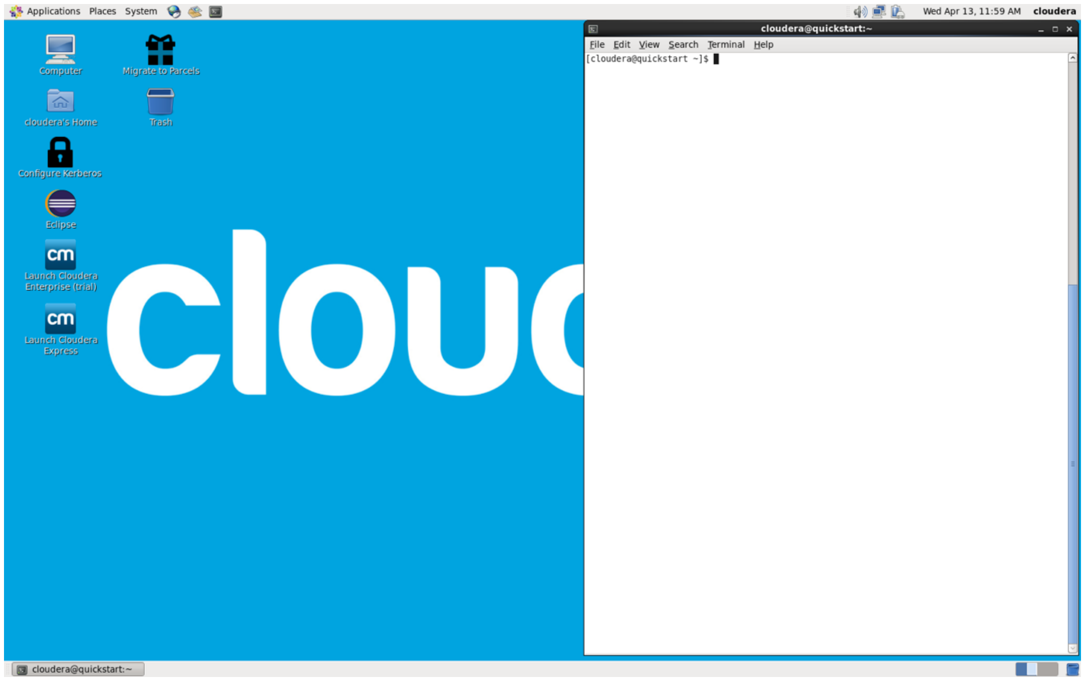
**Question 1 (0.5 pts): Please use your own language to describe the following concepts:**

MapReduce (including Map and Reduce): a programming model, framework, or implementation for processing and generating large data sets using a distributed computing system. It is composed of two main functions, Map, which filters and sorts key-value pairs into a usable form, and Reduce, which aggregates the key-value pairs to create intermediate output. Additionally, the data set must be read by a scanner process, moved from Mappers to Reducers by intermediate steps called shuffling and sorting, and retrieved from the Reducers by a reader process that produces usable output.
Combiner: a less powerful Reducer-like component of MapReduce that is used with a Mapper to optimize intermediate values before moving them on to Reducers. They lower network traffic and lighten the workload of the Reducers.
Hadoop Distributed File System: a program designed to store and process large sets of data using clusters of computers. The program started from research at Google, but is now open-source and managed by Apache. Also called HDFS, it is programmed in Java and built to be fault-tolerant, easy to use, and inexpensive.
Hadoop Name Node: the control center and single point of failure of an HDFS implementation. It tracks the locations and attributes of all files in a cluster and provides the interface to add/copy/move/delete files on the file system. If the NameNode fails, the entire system is offline.
Hadoop Data Node: a server in an HDFS cluster that stores files/data. The DataNode receives its directions for file management from the NameNode, other DataNodes, or applications after they have been directed by the NameNode. DataNodes work together to coordinate data replication and create redundancy in the file system, and periodically update the NameNode on their contents.

**Question 2 (1.5 pts) Hadoop Installation:** Please follow the "MapReduce Programming Platform Installation Instruction" posted in the Blackboard (in the "Lectures" folder) to install Hadoop on your computer. Please report following major steps (capturing screenshots)

- **Part I:** Cloudera MapReduce Installation (0.5 pt)
- **Part II:** First MapReduce Job Task (0.5 pt)
- Pleases report the WordCount task outputs **(**0.5 pt**)**

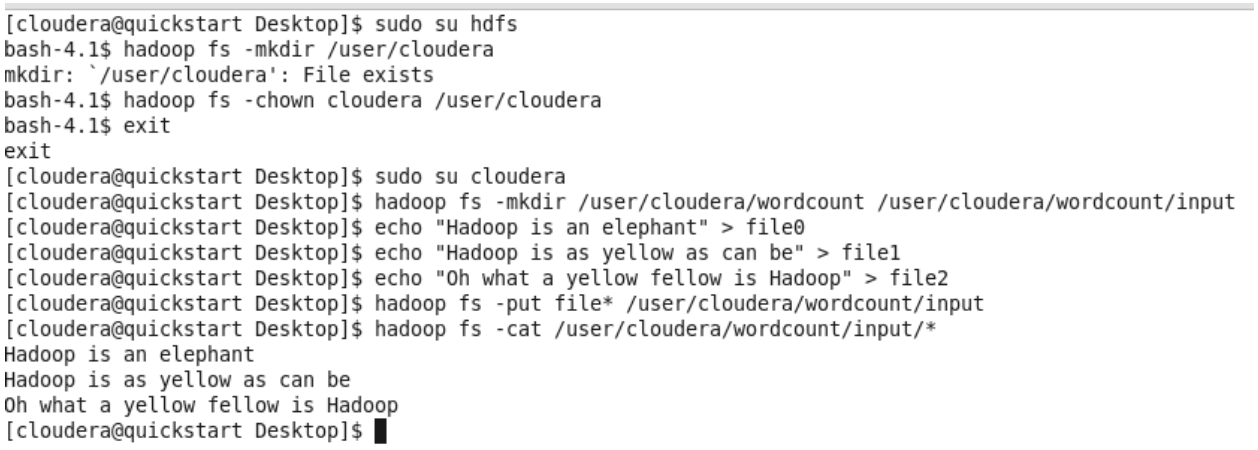
Part I results in the screenshot below. Virtual machine created with VirtualBox, terminal opened.



```
[cloudera@quickstart ~]$
```

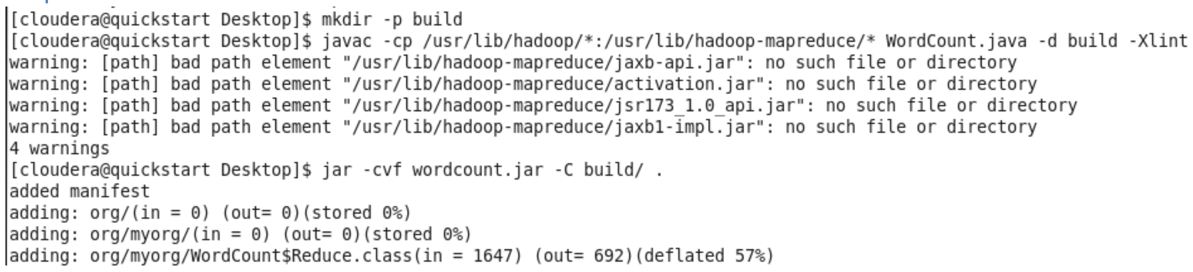
## Part II results in the screenshots below.

## Steps 1 and 2:



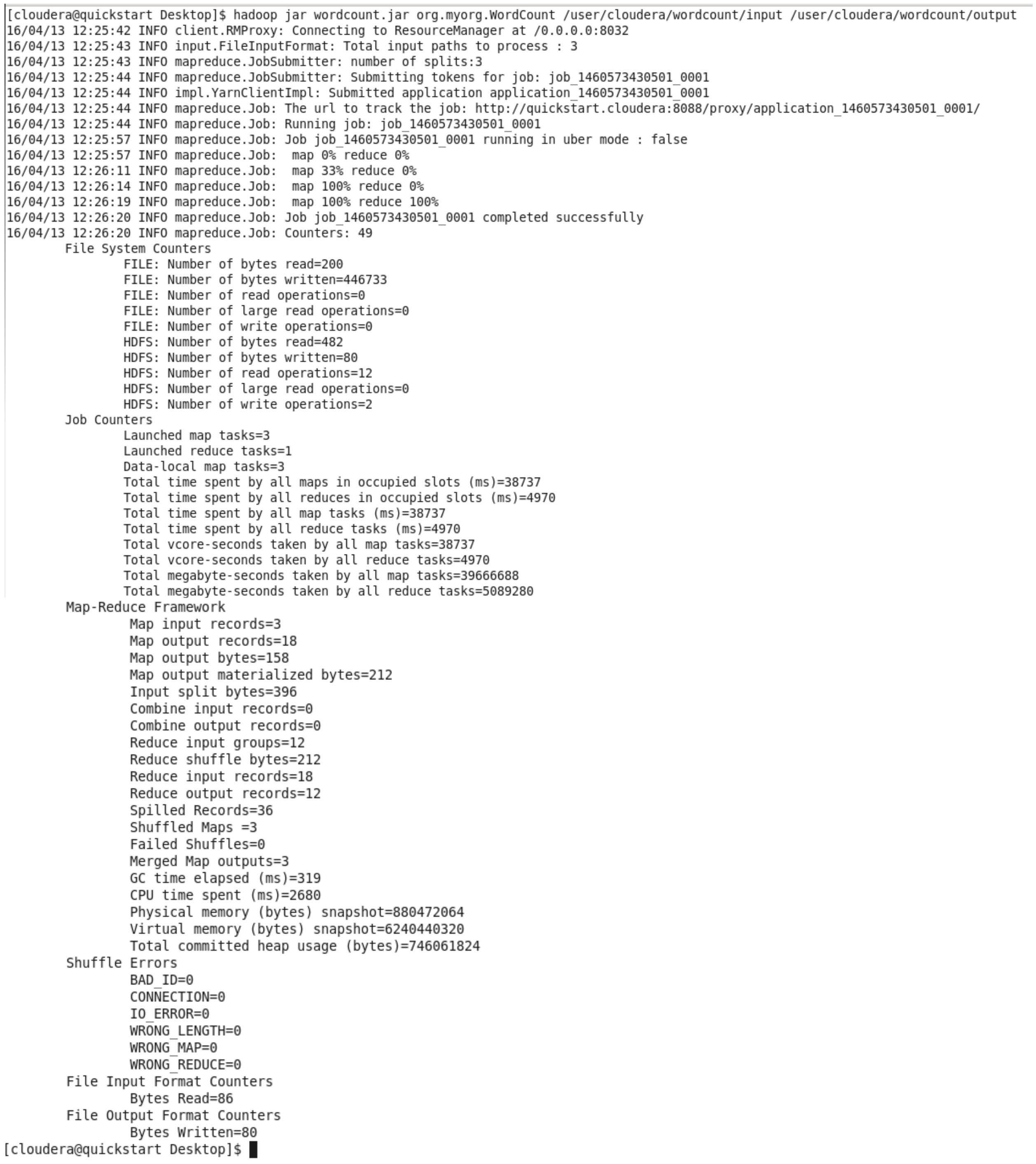
```
[cloudera@quickstart Desktop]$ sudo su hdfs
bash-4.1$ hadoop fs -mkdir /user/cloudera
mkdir: `/user/cloudera': File exists
bash-4.1$ hadoop fs -chown cloudera /user/cloudera
bash-4.1$ exit
exit
[cloudera@quickstart Desktop]$ sudo su cloudera
[cloudera@quickstart Desktop]$ hadoop fs -mkdir /user/cloudera/wordcount /user/cloudera/wordcount/input
[cloudera@quickstart Desktop]$ echo "Hadoop is an elephant" > file0
[cloudera@quickstart Desktop]$ echo "Hadoop is as yellow as can be" > file1
[cloudera@quickstart Desktop]$ echo "Oh what a yellow fellow is Hadoop" > file2
[cloudera@quickstart Desktop]$ hadoop fs -put file* /user/cloudera/wordcount/input
[cloudera@quickstart Desktop]$ hadoop fs -cat /user/cloudera/wordcount/input/*
Hadoop is an elephant
Hadoop is as yellow as can be
Oh what a yellow fellow is Hadoop
[cloudera@quickstart Desktop]$
```

## Steps 3 and 4:



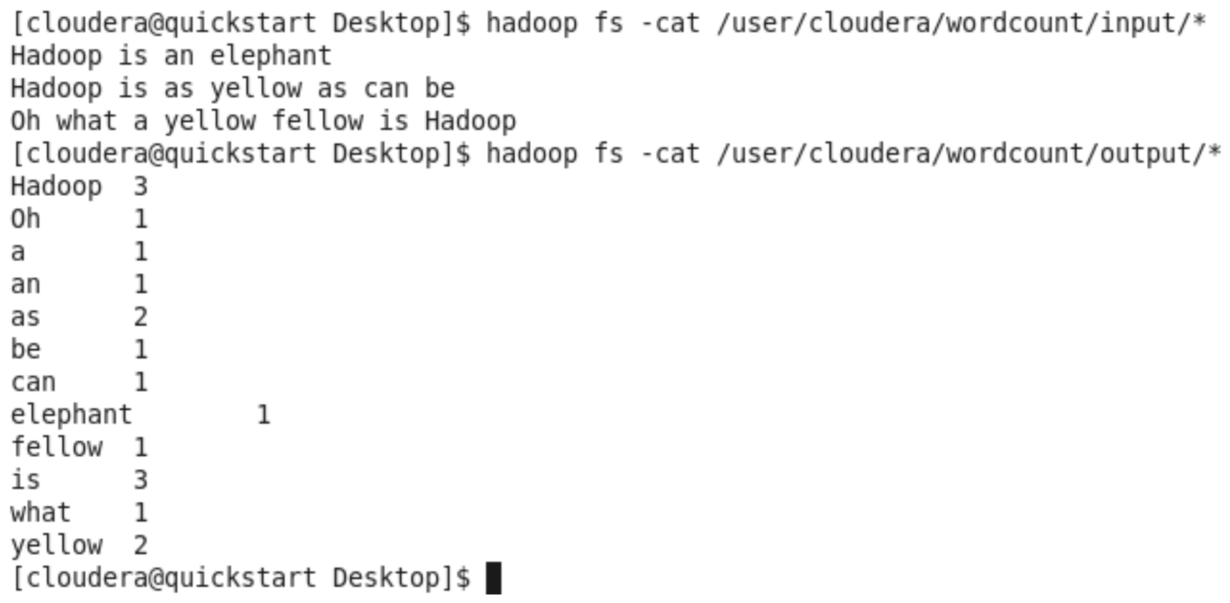
```
[cloudera@quickstart Desktop]$ mkdir -p build
[cloudera@quickstart Desktop]$ javac -cp /usr/lib/hadoop/*:/usr/lib/hadoop-mapreduce/* WordCount.java -d build -Xlint
warning: [path] bad path element "/usr/lib/hadoop-mapreduce/jaxb-api.jar": no such file or directory
warning: [path] bad path element "/usr/lib/hadoop-mapreduce/activation.jar": no such file or directory
warning: [path] bad path element "/usr/lib/hadoop-mapreduce/jsr173_1.0_api.jar": no such file or directory
warning: [path] bad path element "/usr/lib/hadoop-mapreduce/jaxb1-impl.jar": no such file or directory
4 warnings
[cloudera@quickstart Desktop]$ jar -cvf wordcount.jar -C build/ .
added manifest
adding: org/(in = 0) (out= 0)(stored 0%)
adding: org/myorg/(in = 0) (out= 0)(stored 0%)
adding: org/myorg/WordCount$Reduce.class(in = 1647) (out= 692)(deflated 57%)
```

## Output from running .jar file:



```
[cloudera@quickstart Desktop]$ hadoop jar wordcount.jar org.myorg.WordCount /user/cloudera/wordcount/input /user/cloudera/wordcount/output
16/04/13 12:25:42 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/04/13 12:25:43 INFO input.FileInputFormat: Total input paths to process : 3
16/04/13 12:25:43 INFO mapreduce.JobSubmitter: number of splits:3
16/04/13 12:25:44 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1460573430501_0001
16/04/13 12:25:44 INFO impl.YarnClientImpl: Submitted application application_1460573430501_0001
16/04/13 12:25:44 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1460573430501_0001/
16/04/13 12:25:44 INFO mapreduce.Job: Running job: job_1460573430501_0001
16/04/13 12:25:57 INFO mapreduce.Job: Job job_1460573430501_0001 running in uber mode : false
16/04/13 12:25:57 INFO mapreduce.Job:  map 0% reduce 0%
16/04/13 12:26:11 INFO mapreduce.Job:  map 33% reduce 0%
16/04/13 12:26:14 INFO mapreduce.Job:  map 100% reduce 0%
16/04/13 12:26:19 INFO mapreduce.Job:  map 100% reduce 100%
16/04/13 12:26:20 INFO mapreduce.Job: Job job_1460573430501_0001 completed successfully
16/04/13 12:26:20 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=200
                FILE: Number of bytes written=446733
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=482
                HDFS: Number of bytes written=80
                HDFS: Number of read operations=12
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=3
                Launched reduce tasks=1
                Data-local map tasks=3
                Total time spent by all maps in occupied slots (ms)=38737
                Total time spent by all reduces in occupied slots (ms)=4970
                Total time spent by all map tasks (ms)=38737
                Total time spent by all reduce tasks (ms)=4970
                Total vcore-seconds taken by all map tasks=38737
                Total vcore-seconds taken by all reduce tasks=4970
                Total megabyte-seconds taken by all map tasks=39666688
                Total megabyte-seconds taken by all reduce tasks=5089280
        Map-Reduce Framework
                Map input records=3
                Map output records=18
                Map output bytes=158
                Map output materialized bytes=212
                Input split bytes=396
                Combine input records=0
                Combine output records=0
                Reduce input groups=12
                Reduce shuffle bytes=212
                Reduce input records=18
                Reduce output records=12
                Spilled Records=36
                Shuffled Maps =3
                Failed Shuffles=0
                Merged Map outputs=3
                GC time elapsed (ms)=319
                CPU time spent (ms)=2680
                Physical memory (bytes) snapshot=880472064
                Virtual memory (bytes) snapshot=6240440320
                Total committed heap usage (bytes)=746061824
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=86
        File Output Format Counters
                Bytes Written=80
[cloudera@quickstart Desktop]$ █
```

## Step 5 complete, wordcount program run with provided inputs:

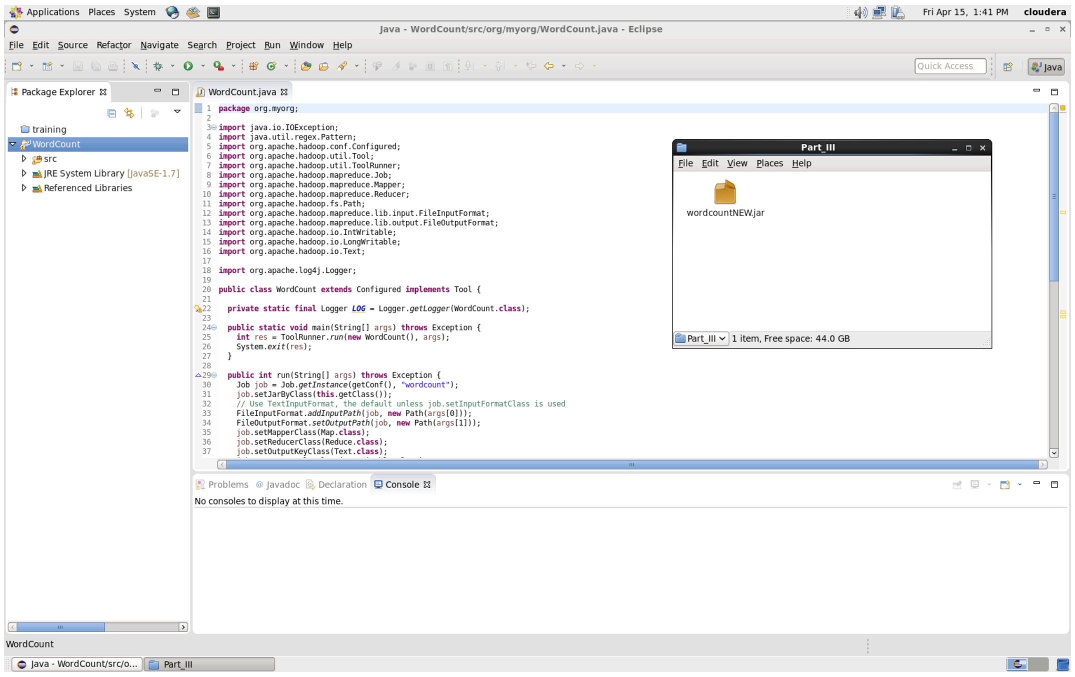


```
[cloudera@quickstart Desktop]$ hadoop fs -cat /user/cloudera/wordcount/input/*
Hadoop is an elephant
Hadoop is as yellow as can be
Oh what a yellow fellow is Hadoop
[cloudera@quickstart Desktop]$ hadoop fs -cat /user/cloudera/wordcount/output/*
Hadoop  3
Oh      1
a       1
an      1
as      2
be      1
can     1
elephant        1
fellow  1
is      3
what    1
yellow  2
[cloudera@quickstart Desktop]$ █
```

**Question 3 (2 pts) Eclipse Hadoop project development:** Please follow the installation instruction Part III (**Part III:** Eclipse MapReduce programming platform) to create a WordCount Eclipse Project. You can use WordCount.java file downloaded from the Cloudera website (please refer to the instruction for details). After that, please report the following major steps (capturing screenshots)

- Report that you have created an Eclipse WordCount project (0.5 pt)
- Report that you can compile the WordCount project and output JAR file (0.5 pt)
- Run WordCound.Jar as a MapReduce tasks, and report the output (1 pt)

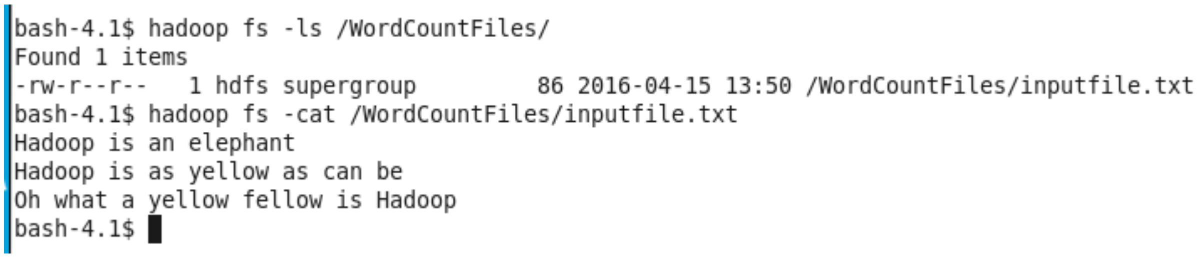
The WordCount project is created in Eclipse and the .jar file is output (called wordcountNEW):


Applications Places System
Fri Apr 15, 1:41 PM cloudera
Java - WordCount/src/org/myorg/WordCount.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Package Explorer
training
WordCount
src
JRE System Library [JavaSE-1.7]
Referenced Libraries
WordCount.java
Part_III
File Edit View Places Help
wordcountNEW.jar
Part_III
1 item, Free space: 44.0 GB
Problems
Javadoc
Declaration
Console
WordCount
Java - WordCount/src/o...
Part_III

```java
package org.myorg;

import java.io.IOException;
import java.util.regex.Pattern;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;

import org.apache.log4j.Logger;

public class WordCount extends Configured implements Tool {

  private static final Logger LOG = Logger.getLogger(WordCount.class);

  public static void main(String[] args) throws Exception {
    int res = ToolRunner.run(new WordCount(), args);
    System.exit(res);
  }

  public int run(String[] args) throws Exception {
    Job job = Job.getInstance(getConf(), "wordcount");
    job.setJarByClass(this.getClass());
    // Use TextInputFormat, the default unless job.setInputFormatClass is used
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);
    job.setOutputKeyClass(Text.class);
```

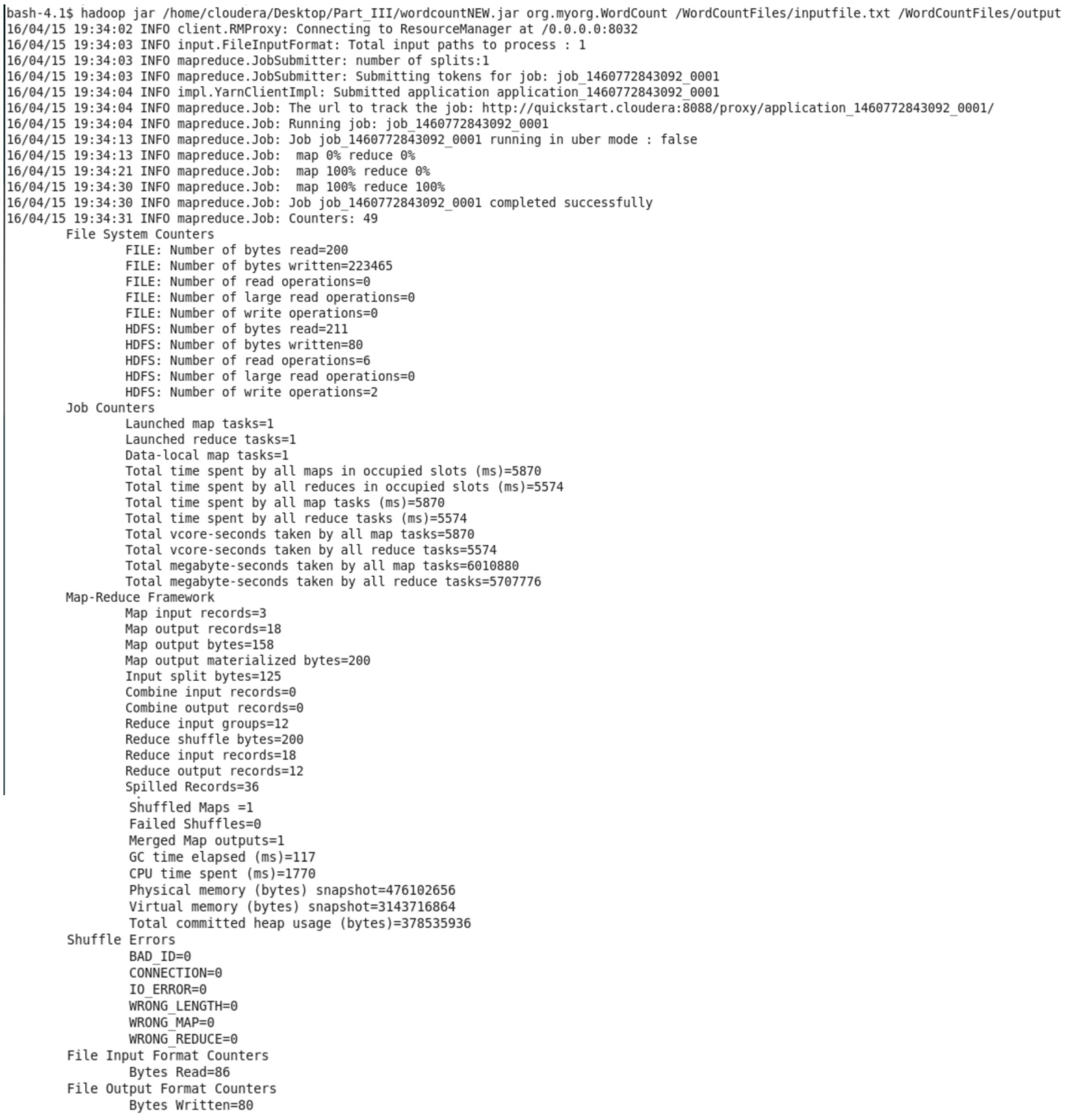
No consoles to display at this time.

## Input file to be used with wordcountNEW.jar is moved into the HDFS:



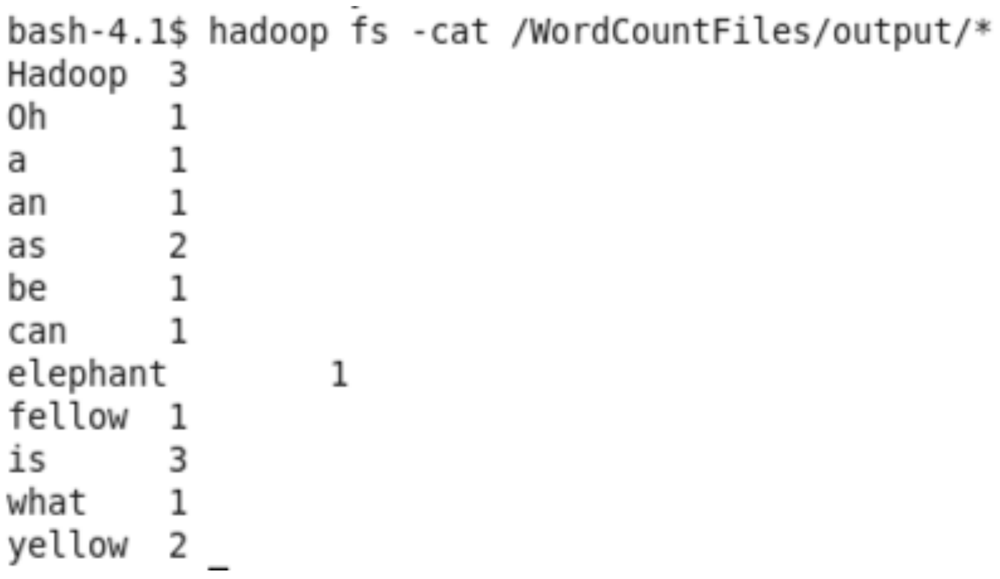
```
bash-4.1$ hadoop fs -ls /WordCountFiles/
Found 1 items
-rw-r--r--   1 hdfs supergroup         86 2016-04-15 13:50 /WordCountFiles/inputfile.txt
bash-4.1$ hadoop fs -cat /WordCountFiles/inputfile.txt
Hadoop is an elephant
Hadoop is as yellow as can be
Oh what a yellow fellow is Hadoop
bash-4.1$ 
```

## The MapReduce task is run via wordcountNEW.jar:



```
bash-4.1$ hadoop jar /home/cloudera/Desktop/Part_III/wordcountNEW.jar org.myorg.WordCount /WordCountFiles/inputfile.txt /WordCountFiles/output
16/04/15 19:34:02 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/04/15 19:34:03 INFO input.FileInputFormat: Total input paths to process : 1
16/04/15 19:34:03 INFO mapreduce.JobSubmitter: number of splits:1
16/04/15 19:34:03 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1460772843092_0001
16/04/15 19:34:04 INFO impl.YarnClientImpl: Submitted application application_1460772843092_0001
16/04/15 19:34:04 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1460772843092_0001/
16/04/15 19:34:04 INFO mapreduce.Job: Running job: job_1460772843092_0001
16/04/15 19:34:13 INFO mapreduce.Job: Job job_1460772843092_0001 running in uber mode : false
16/04/15 19:34:13 INFO mapreduce.Job:  map 0% reduce 0%
16/04/15 19:34:21 INFO mapreduce.Job:  map 100% reduce 0%
16/04/15 19:34:30 INFO mapreduce.Job:  map 100% reduce 100%
16/04/15 19:34:30 INFO mapreduce.Job: Job job_1460772843092_0001 completed successfully
16/04/15 19:34:31 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=200
                FILE: Number of bytes written=223465
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=211
                HDFS: Number of bytes written=80
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=5870
                Total time spent by all reduces in occupied slots (ms)=5574
                Total time spent by all map tasks (ms)=5870
                Total time spent by all reduce tasks (ms)=5574
                Total vcore-seconds taken by all map tasks=5870
                Total vcore-seconds taken by all reduce tasks=5574
                Total megabyte-seconds taken by all map tasks=6010880
                Total megabyte-seconds taken by all reduce tasks=5707776
        Map-Reduce Framework
                Map input records=3
                Map output records=18
                Map output bytes=158
                Map output materialized bytes=200
                Input split bytes=125
                Combine input records=0
                Combine output records=0
                Reduce input groups=12
                Reduce shuffle bytes=200
                Reduce input records=18
                Reduce output records=12
                Spilled Records=36
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=117
                CPU time spent (ms)=1770
                Physical memory (bytes) snapshot=476102656
                Virtual memory (bytes) snapshot=3143716864
                Total committed heap usage (bytes)=378535936
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=86
        File Output Format Counters
                Bytes Written=80
```

The final results are output:



```
bash-4.1$ hadoop fs -cat /WordCountFiles/output/*
Hadoop  3
Oh      1
a       1
an      1
as      2
be      1
can     1
elephant        1
fellow  1
is      3
what    1
yellow  2 _
```

**Question 4 (3 pts) Mapper functions:** In the attached "WordCountLmc.java" and "WordCountGmc.java", the mapper functions create associate array to maintain key-value pair status. The difference is that WordCountLmc uses local in-mapper-combing, and WordCountGmc uses global in-mapper-combing.

- Please modify your Hadoop Project in Question 3, to create a new project "WordCountLmc", which uses local in-mapper-combing to count word frequency. Please use "genesis.txt", "luke.txt", and "kings.txt" as input (place all three files in a folder named "input"), and report the running results of the project. (1 pt)
- Please modify your Hadoop Project in Question 3, to create a new project "WordCountGmc", which uses global in-mapper-combing to count word frequency. Please use "genesis.txt", "luke.txt", and "kings.txt" as input (place all three files in a folder named "input"), and report the running results of the project. (1 pt)
- Please compare running results from three MapReduce Tasks, WordCount, WordCountLmc, and WordCoundGmc. Analyze and report the differences (i.e. Explain the benefits of in-mapper-combining, and explain how local and global in-mapper-combing achieve the efficiency gain) (1 pt)

**Question 5 (1.5 pt) Bigram Counting MapReduce Task:** Given a sentence, a bigrapm denotes a unit consists of two consecutive words of the sentence. For example, given a sentence "I am a student at FAU", there are five bigrams: (I am), (am a) (a student) (student at) (at FAU). Bigrams are used to preserve the context information in the sentence.

Please deign a MapReduce task, which takes "genesis.txt" as the input, and count the frequency of all bigrams (excluding punctuations).

- Please submit your java file [1.0 pt]
- Please use "genesis.txt" as input, and report the final counting results [0.5 pt]