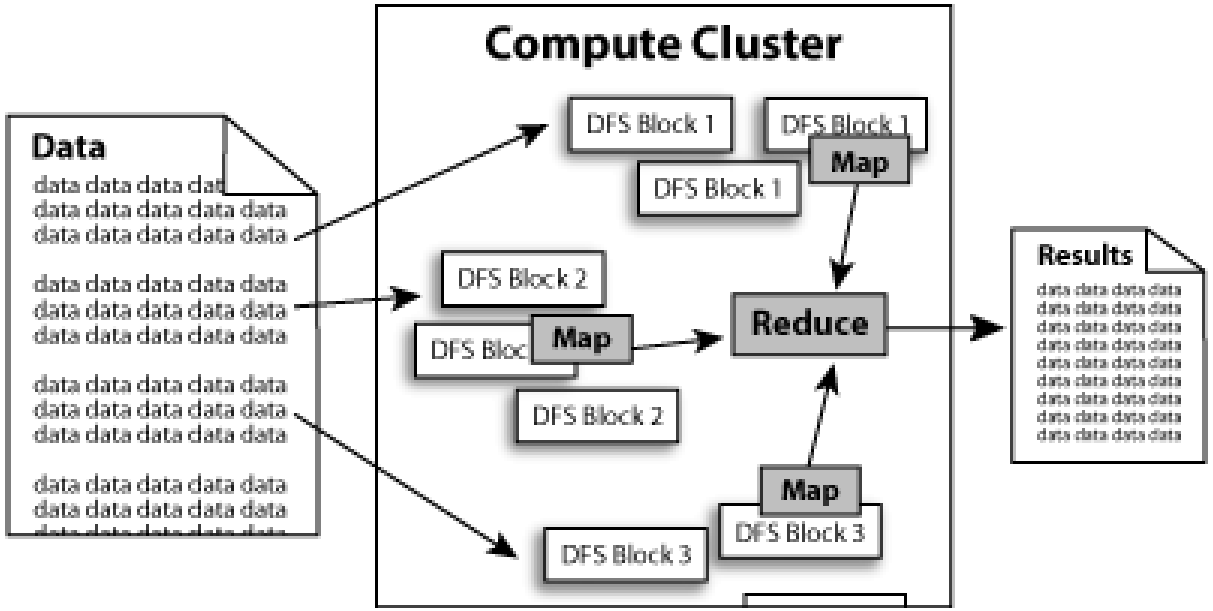
Word Counting Using MapReduce

Sorayut Glomglome

Content

1. Explain MapReduce for word count
2. Hands-on practice for MapReduce

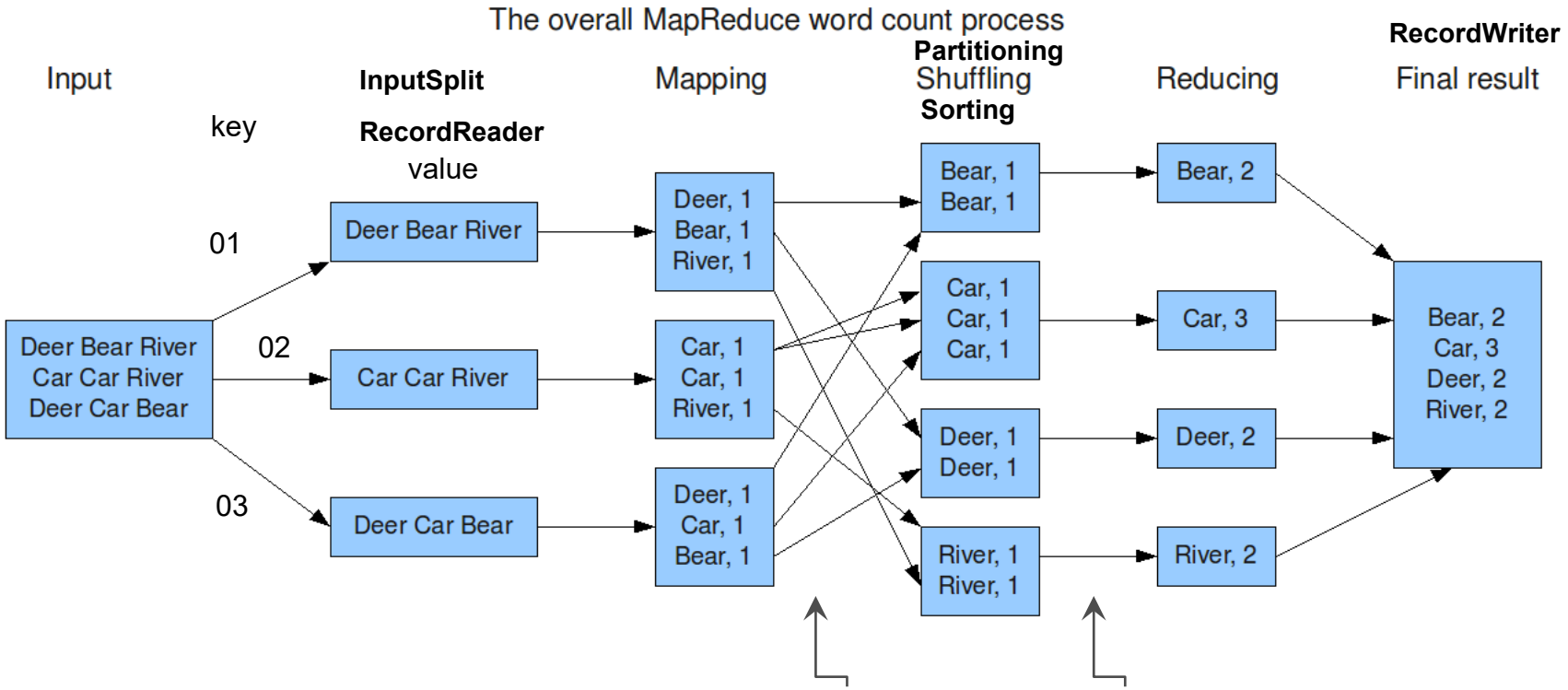
MapReduce on Hadoop




```
map: (K1, V1) -> list(K2, V2))
reduce: (K2, list(V2)) -> list(K3, V3)
```

MapReduce - Word Count

```
map: (K1, V1) -> list(K2, V2))
reduce: (K2, list(V2)) -> list(K3, V3)
```

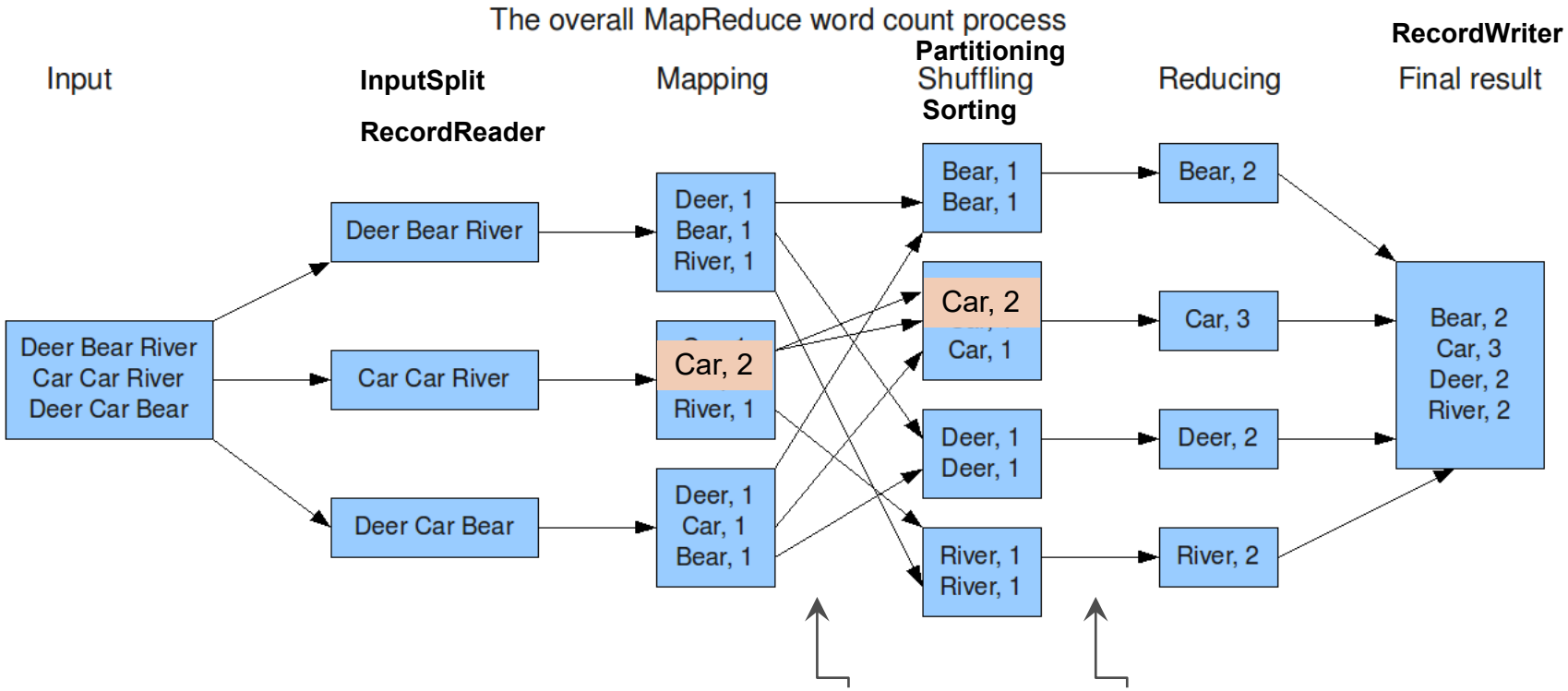


Output in a list of (Key, Value) in the intermediate file

Output in a list of (Key, List of Values) in the intermediate file

MapReduce - Word Count (accelerating)

```
map: (K1, V1) -> list(K2, V2))
reduce: (K2, list(V2)) -> list(K3, V3)
```



Output in a list of (Key, Value) in the intermediate file

Output in a list of (Key, List of Values) in the intermediate file

MapReduce Processing – The Data flow

1. InputFormat, InputSplits, RecordReader
2. Mapper - *your focus is here*
3. Partition, Shuffle & Sort
4. Reducer - *your focus is here*
5. OutputFormat, RecordWriter

InputFormat

InputFormat:	Description:	Key:	Value:
TextInputFormat	Default format; reads lines of text files	The byte offset of the line	The line contents
KeyValueInputFormat	Parses lines into key, val pairs	Everything up to the first tab character	The remainder of the line
SequenceFileInputFormat	A Hadoop-specific high-performance binary format	user-defined	user-defined

InputSplit

An InputSplit describes a unit of work that comprises a single ***map task***.

InputSplit presents a byte-oriented view of the input.

You can control this value by setting the **mapred.min.split.size** parameter in **core-site.xml**, or by overriding the parameter in the JobConf object used to submit a particular MapReduce job.

RecordReader

RecordReader reads <key, value> pairs from an InputSplit.

Typically the RecordReader converts the byte-oriented view of the input, provided by the InputSplit, and presents a record-oriented to the Mapper

Mapper

```
map: (K1, V1) -> list(K2, V2))
reduce: (K2, list(V2)) -> list(K3, V3)
```

The **Mapper** performs the **user-defined logic** to the input a key, value and emits **(key, value) pair(s)** which are forwarded to the **Reducers**.

Partition, Shuffle & Sort

```
map: (K1, V1) -> list(K2, V2))
reduce: (K2, list(V2)) -> list(K3, V3)
```

After the first map tasks have completed, the nodes may still be performing several more map tasks each. But they also begin exchanging the intermediate outputs from the map tasks to where they are required by the reducers.

Partitioner controls the partitioning of map-outputs to assign to reduce task . The total number of partitions is the same as the number of reduce tasks for the job

The set of intermediate keys on a single node is **automatically sorted** by internal Hadoop before they are presented to the Reducer

This process of moving map outputs to the reducers is known as **shuffling**.

Reducer

```
map: (K1, V1) -> list(K2, V2))
reduce: (K2, list(V2)) -> list(K3, V3)
```

This is an instance of **user-provided code** that performs read each key, iterator of values in the partition assigned. The **OutputCollector** object in **Reducer** phase has a method named **collect()** which will collect a **(key, value) output**.

OutputFormat, Record Writer

OutputFormat governs the writing format in **OutputCollector** and **RecordWriter** writes output into HDFS.

TextOutputFormat	**Default; writes lines in "key \t value" form**
SequenceFileOutputFormat	**Writes binary files suitable for reading into subsequent MapReduce jobs**
NullOutputFormat	**generates no output files**

Hands-on practice for MapReduce (Word Count)

Create a Java file

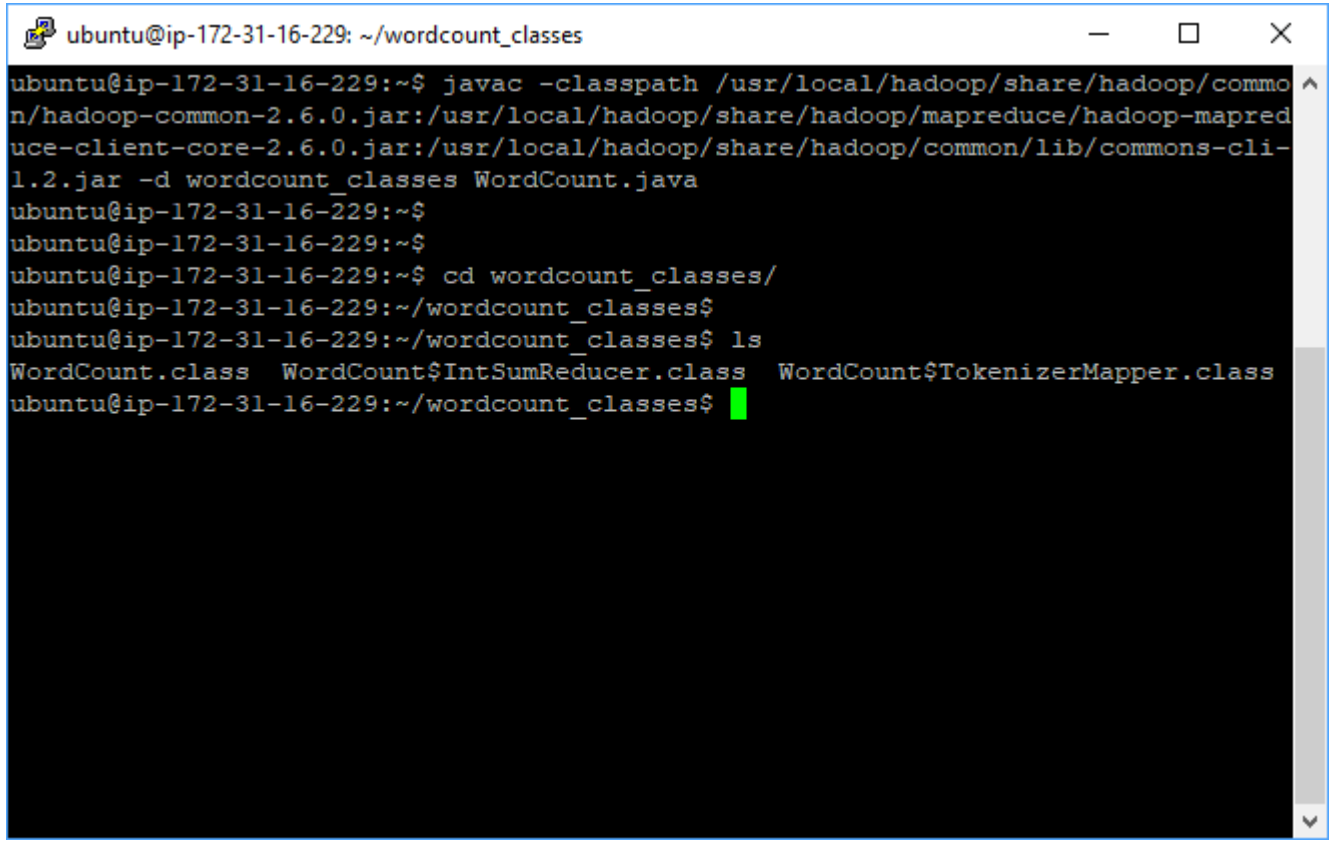
$nano WordCount.java

```
ubuntu@ip-172-31-16-229: ~
ubuntu@ip-172-31-16-229:~$ nano WordCount.java
ubuntu@ip-172-31-16-229:~$
```

```
ubuntu@ip-172-31-16-229: ~
  GNU nano 2.5.3            File: WordCount.java                    Modified

import java.io.IOException;
import java.util.StringTokenizer;
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.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

        public static class TokenizerMapper
                extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

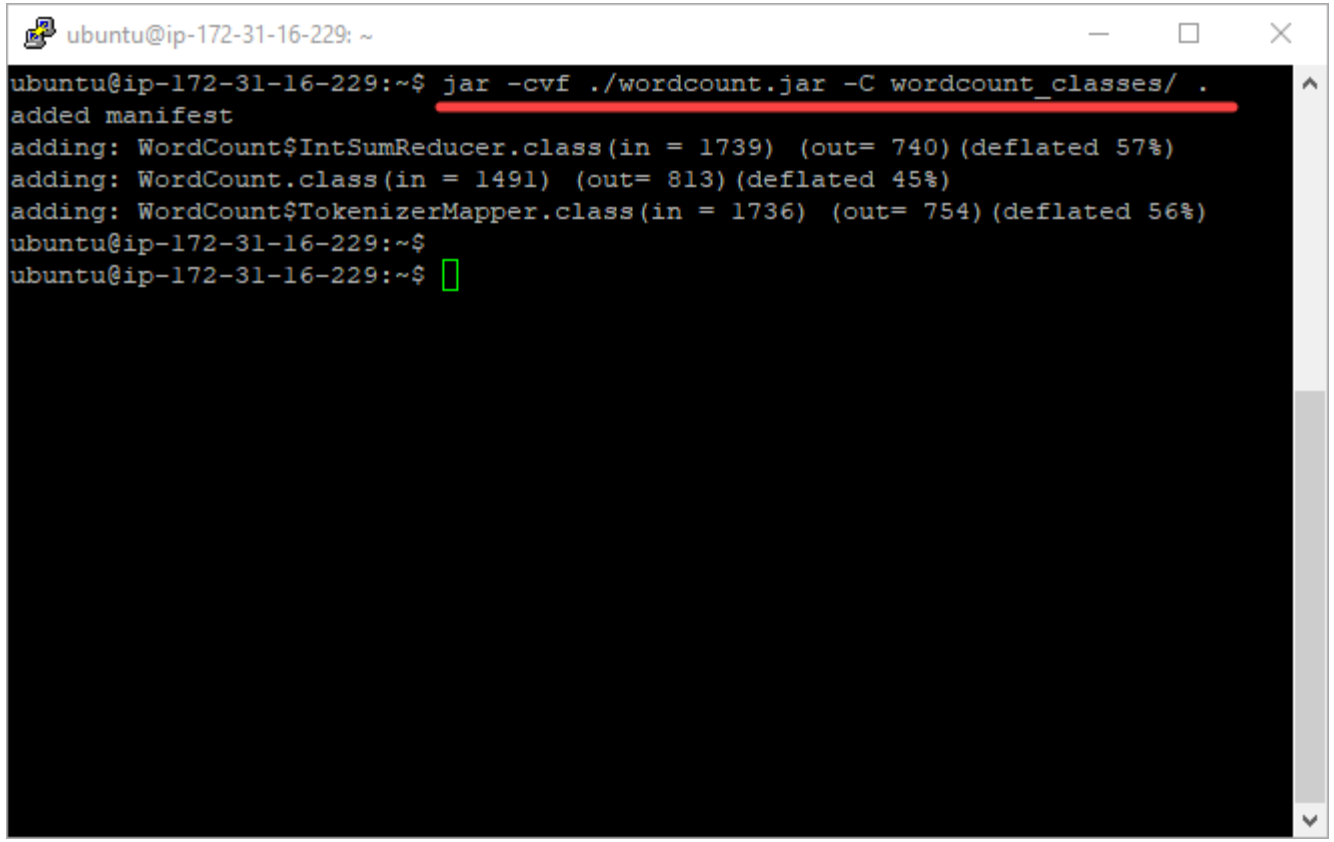
Create a directory for complied code and compile

```
$mkdir wordcount_classes

$javac -classpath /usr/local/hadoop/share/hadoop/common/hadoop-common-2.6.0.jar:/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.6.0.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-cli-1.2.jar -d wordcount_classes WordCount.java
```



Create a Jar file

```
$jar -cvf ./wordcount.jar -C wordcount_classes/ .
```



Don't forget "."!!!

Add a MapReduce Job to YARN

```
$yarn jar ./wordcount.jar WordCount /inputs/* /outputs/wordcount_output_dir01
```

Run all files in directory.

Can't overwrite existing directory.

Use the new one each time.

ubuntu@ip-172-31-16-229: ~

```
ubuntu@ip-172-31-16-229:~$ jar -cvf ./wordcount.jar -C wordcount_classes/ .
added manifest
adding: WordCount$IntSumReducer.class(in = 1739) (out= 740)(deflated 57%)
adding: WordCount.class(in = 1491) (out= 813)(deflated 45%)
adding: WordCount$TokenizerMapper.class(in = 1736) (out= 754)(deflated 56%)
ubuntu@ip-172-31-16-229:~$
ubuntu@ip-172-31-16-229:~$ yarn jar ./wordcount.jar WordCount /inputs/* /outputs
/wordcount_output_dir01
19/02/20 13:21:53 INFO Configuration.deprecation: session.id is deprecated. Inst
ead, use dfs.metrics.session-id
19/02/20 13:21:53 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName
=JobTracker, sessionId=
19/02/20 13:21:53 WARN mapreduce.JobSubmitter: Hadoop command-line option parsin
g not performed. Implement the Tool interface and execute your application with
ToolRunner to remedy this.
19/02/20 13:21:53 INFO input.FileInputFormat: Total input paths to process : 1
19/02/20 13:21:53 INFO mapreduce.JobSubmitter: number of splits:1
19/02/20 13:21:53 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_lo
cal581792515_0001
19/02/20 13:21:53 INFO mapreduce.Job: The url to track the job: http://localhost
:8080/
19/02/20 13:21:53 INFO mapreduce.Job: Running job: job_local581792515_0001
19/02/20 13:21:53 INFO mapred.LocalJobRunner: OutputCommitter set in config null
19/02/20 13:21:53 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hado
```

View the result using command line

```
$hdfs dfs -cat /outputs/wordcount_output_dir01/part-r-00000
```

ubuntu@ip-172-31-16-229: ~

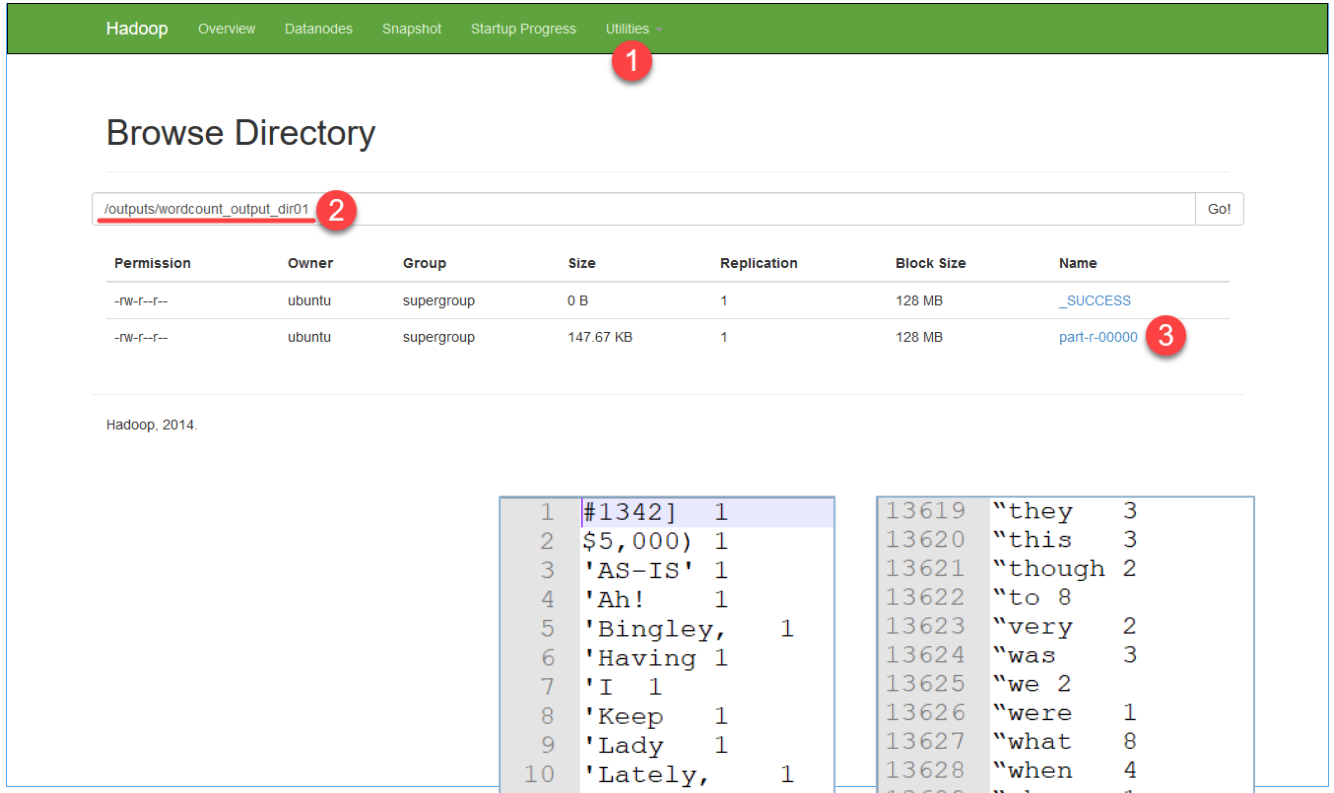
```
ubuntu@ip-172-31-16-229:~$ hdfs dfs -cat /outputs/wordcount_output_dir01/part-r-
00000
```

ubuntu@ip-172-31-16-229: ~

```
"then    1
"there   3
"these   1
"they    3
"this    3
"though  2
"to      8
"very    2
"was     3
"we      2
"were    1
"what    8
"when    4
"where   1
"whether         2
"which   3
"who     2
"why     1
"will    2
"without         1
"yes,    1
"you     10
"your    4
ubuntu@ip-172-31-16-229:~$
```

View the result using web console

http://52.26.15.54:50070




```
#1342]  1
$5,000) 1
'AS-IS' 1
'Ah!    1
'Bingley,   1
'Having 1
'I  1
'Keep   1
'Lady   1
'Lately,    1
'Lydia  1
'Mr.    1
'Oh!    1
'This   1
'Tis    1
'Yes,'  1
'_She_  1
'had    1
'violently  1
'you    1
```

```
"they   3
"this   3
"though 2
"to 8
"very   2
"was    3
"we 2
"were   1
"what   8
"when   4
"where  1
"whether    2
"which  3
"who    2
"why    1
"will   2
"without    1
"yes,   1
"you    10
"your   4
```

Practice for MapReduce (Weather Data)

ftp://ftp.ncdc.noaa.gov/pub/data/noaa/2018

Instruction

HighestDriver.java
HighestMapper.java
HighestReducer.java

1. Import all files to EC2
2. Compile three java files
3. Pack class files to a jar file
4. Input weather data files to hdfs directory
5. Run MapReduce

Don't Forget to STOP or TERMINATE EC2 Instance !!!!