Course: Big Data

*Tutorial 03*

**MapReduce**

## **Simple WordCount MapReduce Job**

### Prepare data for the MapReduce job

1. Create a folder for the job wordcount

bin/hdfs dfs -mkdir wordcount

1. Create a input folder

bin/hdfs dfs -mkdir wordcount/input

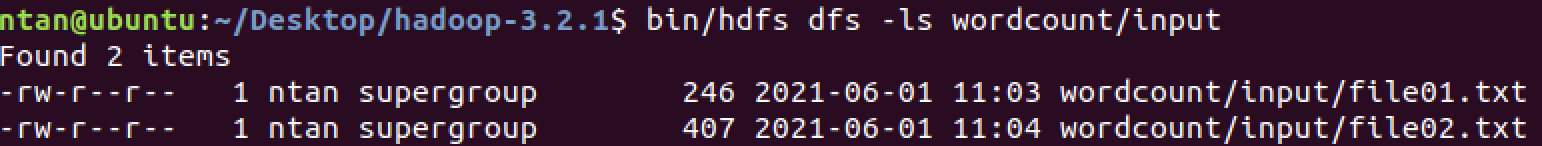
1. Copy two text files to folder input

bin/hdfs dfs -put file01.txt wordcount/input/

bin/hdfs dfs -put file02.txt wordcount/input/

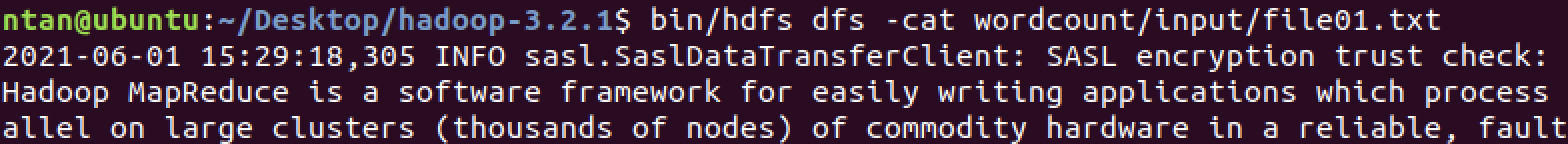
1. Verify copied data

bin/hdfs dfs -ls wordcount/input



1. Show the content of a text file

bin/hdfs dfs -cat wordcount/input/file01.txt



### Create a Java Program for the job

**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();**

**public void map(Object key, Text value, Context context**

**) throws IOException, InterruptedException {**

**StringTokenizer itr = new StringTokenizer(value.toString());**

**while (itr.hasMoreTokens()) {**

**word.set(itr.nextToken());**

**context.write(word, one);**

**}**

**}**

**}**

**public static class IntSumReducer**

**extends Reducer<Text,IntWritable,Text,IntWritable> {**

**private IntWritable result = new IntWritable();**

**public void reduce(Text key, Iterable<IntWritable> values,**

**Context context**

**) throws IOException, InterruptedException {**

**int sum = 0;**

**for (IntWritable val : values) {**

**sum += val.get();**

**}**

**result.set(sum);**

**context.write(key, result);**

**}**

**}**

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

**Configuration conf = new Configuration();**

**Job job = Job.getInstance(conf, "word count");**

**job.setJarByClass(WordCount.class);**

**job.setMapperClass(TokenizerMapper.class);**

**job.setCombinerClass(IntSumReducer.class);**

**job.setReducerClass(IntSumReducer.class);**

**job.setOutputKeyClass(Text.class);**

**job.setOutputValueClass(IntWritable.class);**

**FileInputFormat.addInputPath(job, new Path(args[0]));**

**FileOutputFormat.setOutputPath(job, new Path(args[1]));**

**System.exit(job.waitForCompletion(true) ? 0 : 1);**

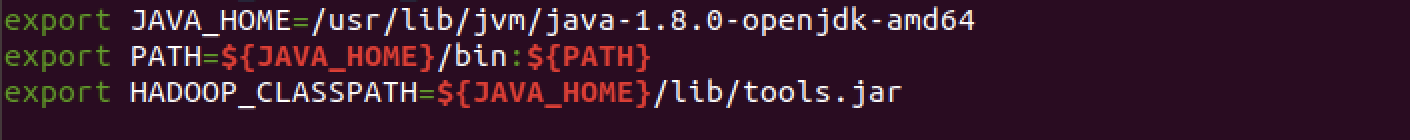
**}**

**}**

### Compile and Execute the job

1. Declare environment variables

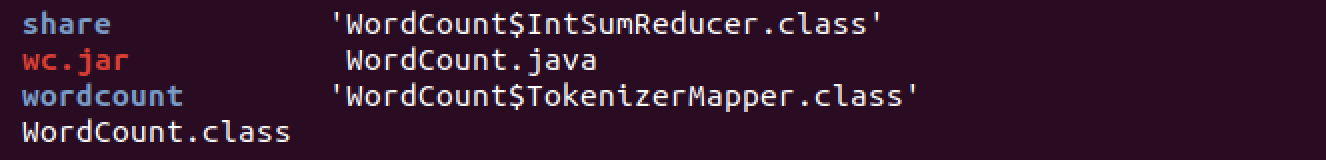
nano etc/hadoop/hadoop-env.sh



1. Compile the WordCount.Java

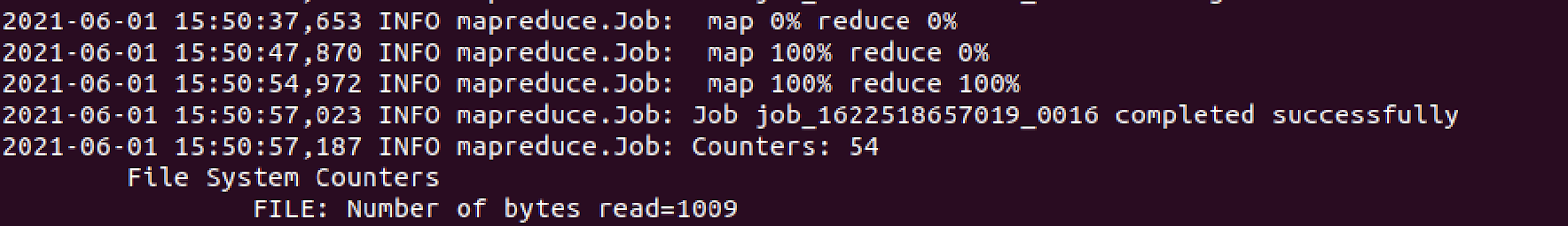
bin/hadoop com.sun.tools.javac.Main WordCount.java

jar cf wc.jar WordCount\*.class



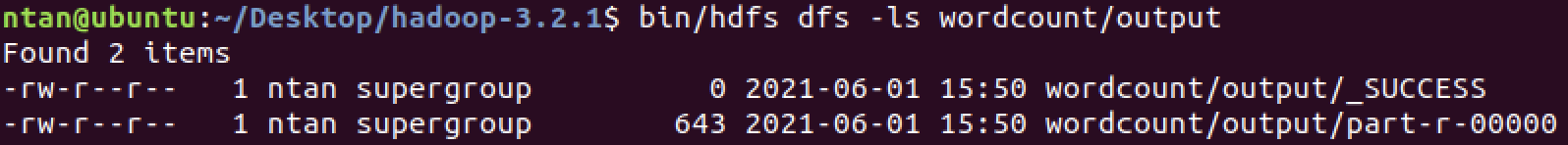
1. Run the application

bin/hadoop jar wc.jar WordCount /user/ntan/wordcount/input /user/ntan/wordcount/output

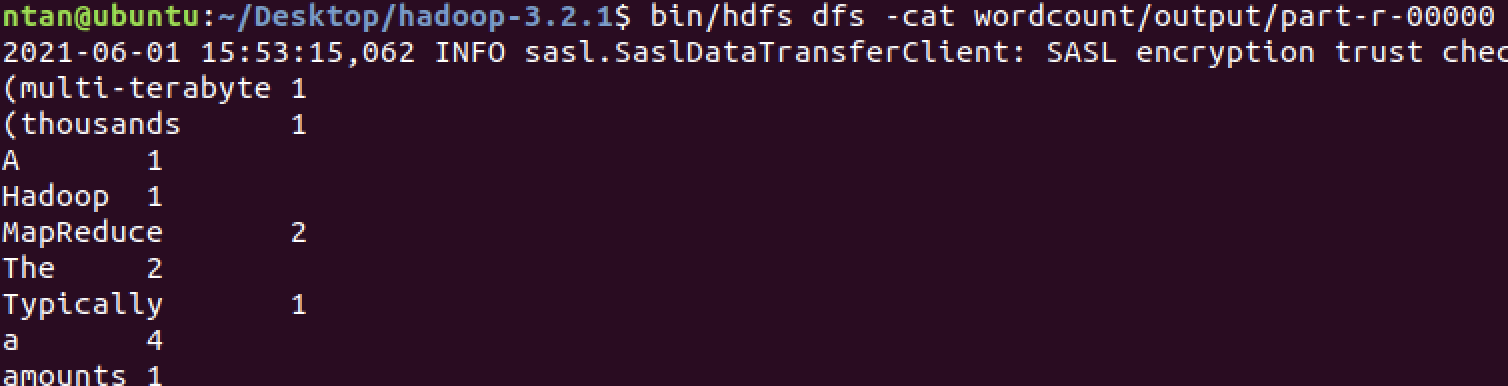


1. Verify the result

bin/hdfs dfs -ls wordcount/output



bin/hdfs dfs -cat wordcount/output/part-r-00000



1. Read more about [Hadoop Command Guides](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/CommandsManual.html) for specifying additional paths, classpaths, archives, etc.

-files

-libjars

-archives

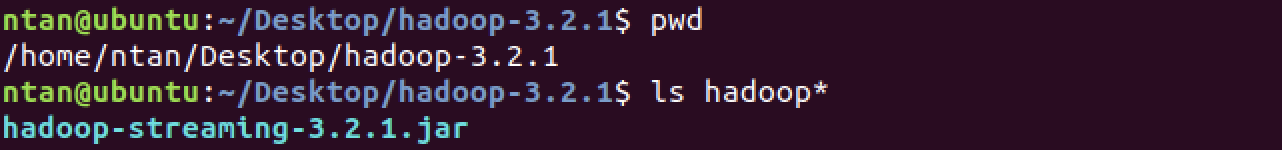
## 

## **Hadoop Streaming for any executables or scripts**

### Basic sample with cat & wc

1. Copy or create a symbolic link to the file hadoop-streaming-3.2.1.jar

ln -s share/hadoop/tools/lib/hadoop-streaming-3.2.1.jar .



1. Run a simple application with cat as a mapper and wc as a reducer

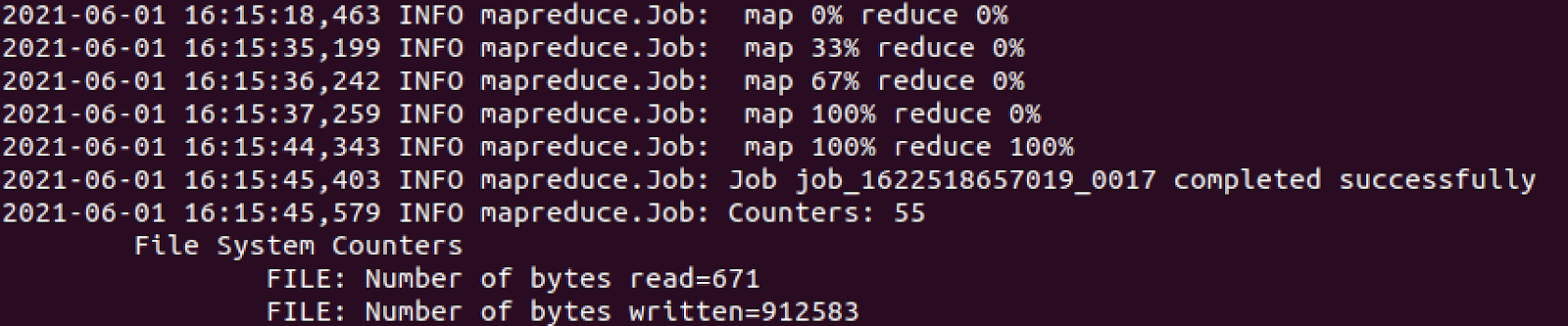
bin/hadoop jar hadoop-streaming-3.2.1.jar \

-input wordcount/input/ \

-output wordcount/output-streaming \

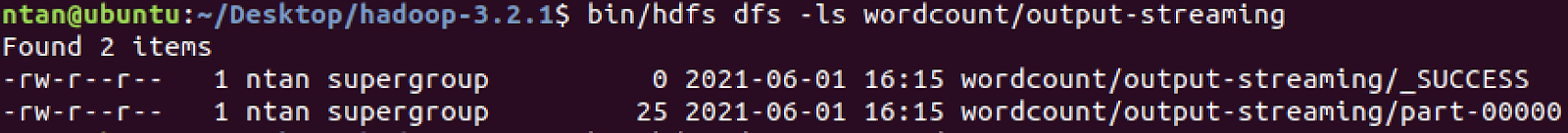
-mapper /bin/cat \

-reducer /usr/bin/wc

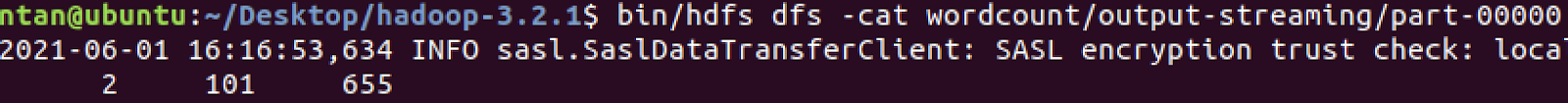


1. Verify the result

bin/hdfs dfs -ls wordcount/output-streaming

****

bin/hdfs dfs -cat wordcount/output-streaming/part-00000



### 

### Using Python Script

1. Create a mapper in Python

nano mapper.py

**#!/usr/bin/python3**

**import sys**

**for line in sys.stdin:**

**for word in line.strip().split():**

**print('%s\t1'%(word))**

1. Create a reducer in Python

nano reducer.py

**#!/usr/bin/python3**

**import sys**

**current\_word = ''**

**current\_count = 0**

**word = ''**

**for line in sys.stdin:**

**try:**

**word, count = line.strip().split('\t', 1)**

**count = int(count)**

**except:**

**continue**

**if word == current\_word:**

**current\_count += count**

**else:**

**if current\_word:**

**print('%s\t%d'%(current\_word,**

**current\_count))**

**current\_word = word**

**current\_count = count**

**print('%s\t%d'%(current\_word, current\_count))**

**if current\_word == word:**

**print '%s\t%s' % (current\_word, current\_count)**

1. Privilege execution permission for the two Python files.

chmod +x mapper.py

chmod +x reducer.py

1. Run a MapReduce job with Python mapper and reducer

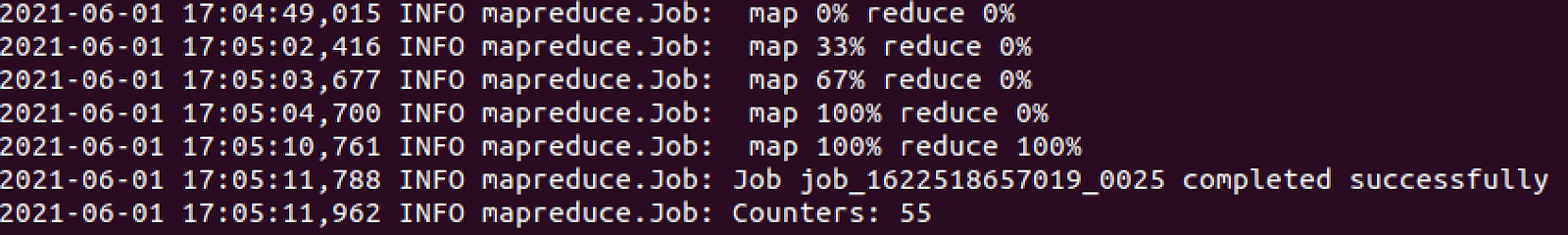
bin/hadoop jar hadoop-streaming-3.2.1.jar \

-input wordcount/input \

-output wordcount/output-py \

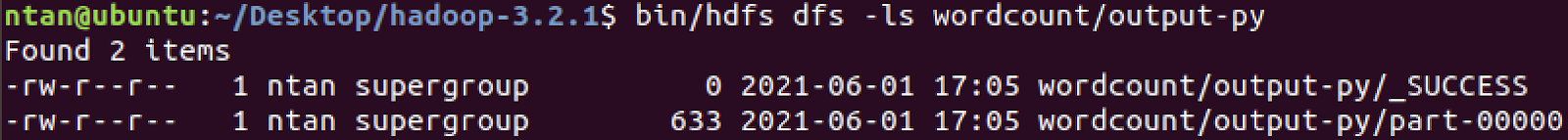
-mapper /home/ntan/Desktop/hadoop-3.2.1/mapper.py \

-reducer /home/ntan/Desktop/hadoop-3.2.1/reducer.py

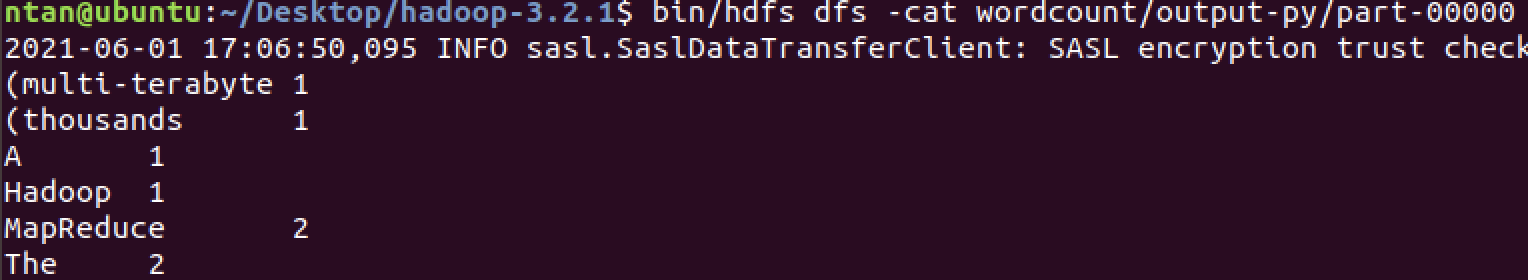


1. Verify result

bin/hdfs dfs -ls wordcount/output-py



bin/hdfs dfs -cat wordcount/output-py/part-00000



# 

# **References**

* <https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Purpose>
* <https://hadoop.apache.org/docs/stable/hadoop-streaming/HadoopStreaming.html#Hadoop_Streaming>
* <https://www.tutorialspoint.com/hadoop/hadoop_streaming.htm>
* <https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/>