python wordcount.py -r hadoop --output-dir out --no-output hdfs://studoop.eecs.qmul.ac.uk/user/fecf3/input

* Check: hadoop fs -ls / out

**Copying to a local file (we are at Lab2 folder, then only this)**

**-hadoop fs -copyToLocal <hdfsfile> <localdestination>**

* bash-4.2$ hadoop fs -copyToLocal out

Check these files, and answer the following two questions:

1. **Can you find any duplicate word among these part- files? Can you explain why?**

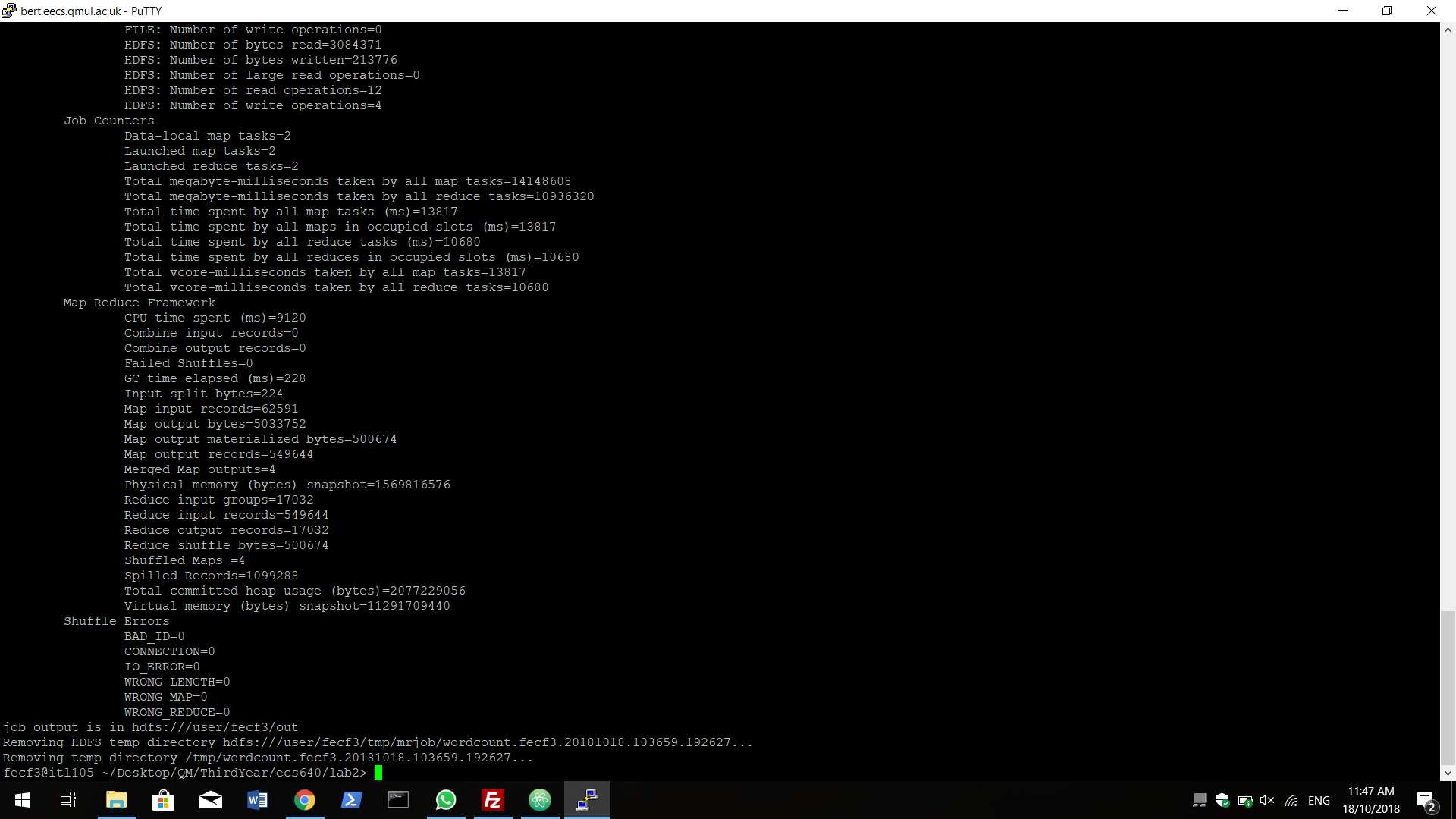
No, I can’t. It is because the part-files are results from the reducers and these reducers receive the values already shuffle in a way that the same words are sent to the same reducer. On startup, the NameNode enters a special state called Safemode, Replication of data blocks does not occur when the NameNode is in the Safemode state.

After retrieving the output you should remove the output folder through the -rm -r command. For example**, in order to remove the out folder** from your HDFS home, type the following command:

hadoop fs -rm -r out

**2. How many Reducers have been involved in the work?**

There have been involved 2 reducers as it can be seen in the graph.



**3. Is there some clear pattern to decide the part file in which each word will appear?**

To determine which reducer instance will receive which intermediate keys and vaues, the mapper must determine for all of its outputs(key,values) pairs which reducer will receive them.

THere is a default partitioner in Hadoop called HashPartitioner which has a method called getPartition. It takes key.hashCode() & INteger.MAX\_VALUE and finds the moduls using the number of reduce tasks.

Here is the code:

public class HashPartitioner<K, V> extends Partitioner<K, V> { public int getPartition(K key, V value, int numReduceTasks) { return (key.hashCode() & Integer.MAX\_VALUE) % numReduceTasks; } }

To create a custom partitioner, you would extend Partitioner, create a method getPartition, then set your partitioner in the driver code (job.setPartitionerClass(CustomPartitioner.class);). This is particularly helpful if doing secondary sort operations, for example.

**Can you see a clear pattern on how Hadoop partitions the keys among multiple reducers? Does it make easier or harder now the problem of manually retrieving information about a specific key? e.g. How many times the word Sherlock appears in the text?**

Sherlock : 345

/data/gutenberg. This is going to take longer to execute, as you are processing a file of size 400MB

python wordcount.py -r hadoop --output-dir out --no-output hdfs://studoop.eecs.qmul.ac.uk/data/gutemberg

Merge folder: 9.9 MBs

**4. How many unique words did we process this time?**

# this class will define two additional methods: the mapper method goes here

def mapper(self, \_, line):

words = WORD\_REGEX.findall(line)

for word in words:

yield (word.lower(), 1)

#and the reducer method goes after this line

# def combiner(self, word, counts):

# x = sum(counts)

# if(x == 1):

# yield("Total", x)

def reducer(self, a, x):

dic = {}

for word in a:

if word not in dic.keys():

dic[word] = 1

yield ("total", sum(dic.values()))

# yield (“total”, sum(x)) # This only if combiner active

"Total" 678956

**5. How many Map tasks does your MapReduce job have? Can you explain the difference with the Sherlock job? How many are actually local mappers (i.e. they are initially collocated with the data they have to process)?**

* **Map tasks does your MapReduce job have(gutemberg):** 4
  + Local mappers: 4
* **Map tasks does your MapReduce job have(Sherlock):** 2
  + Local mapper: 2