CS 8001 Big Data

HW #2: Word Count on Spark (10 points + 2 bonus points)

Jing Su

js929@mail.missouri.edu

In this exercise, you will write Python code on Spark that outputs the 20 most common words in the Complete Works of William Shakespeare retrieved from Project Gutenberg and the number of occurrence of each of them.

Part I:

1) A brief description of your Spark environment, your code structure, and the execution process of the code.

Spark Environment: AWS EMR (1 node), Hadoop 2.6.0, Spark 1.6.0

Code Structure:

1. Create a RDD.
2. Filter the RDD.
3. Transform the RDD.
4. Take actions on the RDD.

Execution Process:

1. Put Sonnets.txt on Hadoop.

```
[hadoop@ip-172-31-29-16 ~]$ hadoop fs -put Sonnets.txt /user/hadoop/Sonnets.txt
```

2. Run sparktop20.py.

```
[hadoop@ip-172-31-29-16 ~]$ spark-submit sparktop20.py Sonnets.txt
```

2) Output of your code (20 most common words and their counts).

```
16/02/01 03:55:07 INFO DAGScheduler: Job 0 finished: takeOrdered at /home/hadoop
/sparktop20.py:23, took 4.668151 s
[(u'the', 27350), (u'and', 26483), (u'i', 20527), (u'to', 18982), (u'of', 17965)
, (u'a', 14491), (u'you', 13537), (u'my', 12386), (u'that', 10991), (u'in', 1084
3), (u'is', 9517), (u'not', 8631), (u'for', 8164), (u'with', 7932), (u'me', 7706
), (u'it', 7613), (u'be', 7012), (u'this', 6799), (u'your', 6799), (u'his', 6760
)]
Execution time: 24.2937600613
```

3) Execution time of your code in seconds.

```
Execution time: 24.2937600613
```

4) Comparison of the implementation and results of this exercise with those of the HW1

Running on Spark spends much more time than running by Python (1s) directly.

5) Your Python code with appropriate comments.

```
from __future__ import print_function
import sys
from operator import add
from pyspark import SparkContext
import re
import time

#Using regular expression to remove punctuation in the file.
def removePunctuation(text):
    return re.sub(r'[^a-z0-9\s]','', text.lower().strip())

if __name__ == "__main__":
    start = time.time()
    #Print error message when the command is wrong.
    if len(sys.argv) != 2:
        print("Usage: wordcount <file>", file=sys.stderr)
        exit(-1)
    #Create a RDD
    sc = SparkContext(appName="PythonWordCount")
    #Split the RDD into 8 partitions and remove punctuation.
    lines = (sc.textFile(sys.argv[1], 8).map(removePunctuation))
    #Transform the RDD into Key-Value pairs.
    counts = lines.flatMap(lambda x: x.split()) \
                  .map(lambda x: (x, 1)) \
                  .reduceByKey(add)
    #Get top 20 words in descending order.
    print(counts.takeOrdered(20,key = lambda x:-x[1]))
    end = time.time()
    print("Execution time:",end-start)
    #Shut down the SparkContexts
    sc.stop()
```

Part II:

1) A brief description of your Spark environment and the execution process of the code.

Spark Environment: AWS EMR (4 nodes), Hadoop 2.6.0, Spark 1.6.0

Execution Process:

1. Create AWS EMR.
2. Put Sonnets.txt on Hadoop.

```
[hadoop@ip-172-31-29-16 ~]$ hadoop fs -put Sonnets.txt /user/hadoop/Sonnets.txt
```

3. Run sparktop20.py.

```
[hadoop@ip-172-31-29-16 ~]$ spark-submit sparktop20.py Sonnets.txt
```

3) Execution time of your code in seconds.

```
Execution time: 39.3881499767
```

4) Comparison of the execution result with that in Part I

The execution time is more than that of running on a single node.