COSC 6339 Big Data Analytics-Assignment2

-Sreekar Reddy Jammula (1422539)

Problem description:

The current assignment is to write the python scripts for Apache Spark. The tasks are divided into three parts as below:

1. **WordCount-**To count the occurrences of words in a book on a per-book basis and compare the results with those of Assignment1.
2. **pyspark.ml. feature-** To count the tf-idf values for the unigram and bigrams using the pyspark.ml.feature p ackage of Mlib library of Spark. Find the execution time using 5,10 and 15 reducers.
3. **Word2Vec-**Find the feature vectors of words using the word2vec class of Mlib library

Input Data Set: A large input data set consisting of 216 books present in the directory /cosc6339_s17/books-longlist/ has been used for this experiments

Outputs: All the outputs have been stored in the /bigd12/ Hadoop dfs directory

Solution Strategy:

The solution strategy is as follows:

1. Task1: For task1, the strategy is similar to the earlier assignment. It simply involves tokenizing the words via the split() function and appending the filename to it. Then each token is mapped to the value 1 using the lambda expression and the map() function. This is then reduced using the reduceByKey() function to produce the word counts on per book basis. A detail understanding of this strategy can be obtained by reading through the comments in the code.
2. Task 2: For task 2, the strategy is to write two separate programs for unigrams and bigrams. Similar to task 1, the words are split. They are then converted into dataframes having two values- the bookname and word. These dataframes are treated as unigrams using the Ngram() function call where n value is 1.These data frames are then transformed using the HashingTF and IDF() functions to produce tf-idf values. The results are saved as text files in the specified output folder. For bigrams, the approach is same except that n value is 2 in Ngram() function call.

3. Task3: The strategy for task 3 is to create an instance of the class Word2Vec and create a Word2Vec model for the input data. To find out the synonyms an array of predefined words is taken. These words have been selected based on three criterias:

    1.They are the most popular English words.

    2.They are known to have a lot of synonyms

    3.There word count in the books is very high.

The words are: amazing,answer,himself,through,without,thought,nothing,another,something,because
Using the findSynonyms() function, the synoyms are printed directly onto the screen.

Programming Environment:

Apache Spark: Apache Spark allows us to easily create BigData applications on the cluster. It is generally considered to be faster than map reduce programming model and is very much useful for large scale applications.

1. It has API's defined for multiple languages and can working with a variety of filesystems and databases. However, for this experiment, we confine ourselves to the python API called PySpark and the Hadoop File System.
2. Spark treats the data in terms of RDD (Resilient Distributed Datasets) which are then manipulated using the transform and action operations.
3. Another main reason for using spark is that it offers powerful graph processing and Machine learning libraries.

PySpark: Pyspark is the python interface for Apache Spark. I

Executing the code:
To execute the the programs follow the following command :
```
spark-submit --master yarn --num-executors <<enter number of executors here>>   <<enter program name here>>
```
Example:
```
spark-submit --master yarn --num-executors 5   asst2_part3.py
```

Results and Observations:

Task 1: The asst2.py program has been executed on 2, 5 and 10 number of executors. The execution time for these have been observed and tabled as below. Please note that the execution times are influenced by various factors such as number of other executions running on the cluster, availability of nodes etc.,

Number of executors = 2:

	Trial 1	Trial 2	Trial 3	Average
Execution Time (in secs)	108	86	95	96.33

Comments: Not much variation in three trials. The average for three trials is taken.

Number of executors = 5:

	Trial 1	Trial 2	Trial 3	Average
Execution Time (in secs)	85	105	93	94

Comments: Not much variation in three trials. The average for three trials is taken.

Number of executors = 10:

	Trial 1	Trial 2	Trial 3	Average
Execution Time (in secs)	157	139	177	157

Comments: Not much variation in three trials. The average for three trials is taken.

Observation:

1. As we observe keenly, the average time decreases when we have 5 executors than 2 . Increasing number of executors helped.
2. When we involve 10 executors, the execution time increases. This is because of parallelization overheads.
3. The execution times described above may be ambiguous. They may involve several factors like the availability of resources, the scheduling of the jobs, the number of partitions. It may always not be true that increasing the number of reducers will decrease the execution time.
4. Achieving the minimum execution time is dependent on a lot of parallelization factors like number of cores available, amount of memory available, number of partitions in RDD etc., and not just on increasing the number of executors.

Comparison with assignment 1

Number of reducers	Avg-Assignment1	Avg-Assignment2
2	155	96.33
5	107	94
10	130	157

Observation: As we observe, the Apache Spark executes the application faster tha Hadoop Map Reduce in most of the cases. The execution time for 10 executors in Spark is more than that of Hadoop. This may be due to some amount of ambiguity as described above.

Task 2: The execution times for task2 -unigrams for 5,10 and 15 executors is as below. Please note that the execution times are influenced by various factors such as number of other executions running on the cluster, availability of nodes etc., Filename: asst2_part2_1.py

Number of executors =5

	Trial 1	Trial 2	Trial 3	Average
Execution Time (in secs)	120	140	132	130

Comments: Not much variation in three trials. The average for three trials is taken.

Number of executors = 10:

	Trial 1	Trial 2	Trial 3	Average
Execution Time (in secs)	93	117	108	106

Comments: Not much variation in three trials. The average for three trials is taken.

Number of executors = 15

	Trial 1	Trial 2	Trial 3	Average
Execution Time (in secs)	102	128	151	127

Comments: Not much variation in three trials. The average for three trials is taken.

Observation: As in part1, the speed increases for 10 executors when compared to 5. However, the time taken for 15 executors is more with the reason being the same.

Task2-Bigrams: asst2_part2_2.py file has to be executed

Number of executors =5

	Trial 1	Trial 2	Trial 3	Average
Execution Time (in secs)	125	155	133	137

Comments: Not much variation in three trials. The average for three trials is taken.

Number of executors = 10:

	Trial 1	Trial 2	Trial 3	Average
Execution Time (in secs)	135	120	128	137

Comments: Not much variation in three trials. The average for three trials is taken.

Number of executors = 15

	Trial 1	Trial 2	Trial 3	Average
Execution Time (in secs)	198	196	202	198

Comments: Not much variation in three trials. The average for three trials is taken.

Observation: The time for executing the bigrams is more than that of trigrams.

Task 3: The asst2_part3.py program has been executed on 5, 10 and 15 number of executors. The execution time for these have been observed and tabled as below. Please note that the execution times are influenced by various factors such as number of other executions running on the cluster, availability of nodes etc.,

Number of executors = 5:

	Trial 1	Trial 2	Trial 3	Average
Execution Time (in secs)	310	300	336	315

Comments: Not much variation in three trials. The average for three trials is taken.

Number of executors = 10:

	Trial 1	Trial 2	Trial 3	Average
Execution Time (in secs)	348	260	341	316

Comments: Not much variation in three trials. The average for three trials is taken.

Number of executors= 15:

	Trial 1	Trial 2	Trial 3	Average
Execution Time (in secs)	398	353	370	373

Comments: Not much variation in three trials. The average for three trials is taken.

Observations: The execution times are expected to follow the trend as in part1. However, due to several reasons, there may be ambiguity in the readings of 10 executors. Overall, task 3 is the most compute intensive as it takes the most amount of time to execute

Resources Used:

These experiments have been performed on the Whale cluster which is under the supervision the parallel software technologies laboratory. The cluster has been accessed through a secured SSH connection from the MobaXterm SSH client.

Given below are the details about the cluster:

Total number of nodes: 57 (whale-001 to whale-0057)

Processor Details: 2.2 GHz quad-core AMD Opteron processor (consisting of 4 cores each)- 2 in number(8 cores total)

Main memory: 16GB

Network Interconnect:

1. 144 port 4xInfiniBand DDR Voltaire Grid Director ISR 2012 switch

2. 48 port HP GE switch- 2 in number

Storage: 20 TB Sun StorageTek 6140 array (/home shared with shark and crill clusters)