HW1 – SPARK AND MAP Reduce Assignment

Rahul Pandey

Github:
https://github.com/2020-F-CS6240/homework-1-mapreduce-rahulpandeycs
https://github.com/2020-F-CS6240/homework-1-spark-rahulpandeycs

1. MAP REDUCE IMPLEMENTATION

1 a) Psuedo Code for Map Reduce

Assume “line” be the input row of the document provided to each map task. This function is Called by each map task for each input row.

```
map(inputLine line)
  for each input line
  (follower, following) = line.split(“,”)
  emit( following, 1)

reduce(followerKey key, values [v1,v2…])
  sum = 0
  for each value in values
    sum = sum + value
   emit(followerKey, sum)
```

1. b) Explanation:

1. The Program is input file edges.csv. The file contains data in the format (userId1, userId2) where *userId1* is the user following the user with *userId2.*
2. The task is to compute the number of followers for each user.
3. The input file is split into smaller chunks and provided to respective map tasks.
4. Each of these map tasks then operate on the input file and calls map function as shown in pseudoCode for each row as input.
5. As each row contains data in format (userId1, userId2) the data is first split by (“,”).
6. After the split we have two user ids , userId1: Follower and userId2 : user being followed.
7. Now we want to count the number of followers for userId2, hence we will emit 1 as the output of this map task, specifying userId2(following) as the key. This ensures that we emit 1 for each time the userId2 is being followed.
8. Now after all the map tasks are completed the output values are shuffled and grouped by the specified key I.e userdId2 in our case.
9. This acts as an input to the reducer task.
10. The reducer gets the input as key(userIds2) and values(List of values emitted from map step associated with this key).
11. These values need to be added in order to return the total count of the followers for this user whose key is specified, hence we iterate over all the values and keep adding them to sum.

12. After complete iteration we emit the total sum value associated with its key.
13. The output will be in format: (userIds2, sum).

2. SPARK SCALA IMPLEMENTATION

2. a) Pseudo Code for Spark

initialize sc as SparkContext with the set of configuration. The sparkContext is use to import the File into the memory. The input row is passed and each row is split into different userIds to get the person being followed using getRDDValue(). After this step we have (personBeingFollowed,1) being emitted which is further reduced using reduceByKey method Which groups values by there key and reduce them using the past function i.e sum in this case.

```
func(inputFile, outputFile)
  textFileRow = sc.ReadTextFile(inputFile)
  rddValue = getRDDValue(textFileRow)
  numberOfFollowers = rddValue.reduceByKey((value1, value2) => value1+value2)
  numberOfFollowers.saveToFile(outputFile)

getRDDValue(textFileRow)
  personBeingFollowed = textFileRow.split(“,”)(1)
  rddValue = map(personBeingFollowed,1)
  return rddValue
```

2. b) Spark Debug String output:

Run 1:

```
  20/09/24 16:22:13 INFO root: The value of count(20) ShuffledRDD[4] at reduceByKey at
twitterFollowerCount.scala:28 []
 +-(20) MapPartitionsRDD[3] at map at twitterFollowerCount.scala:27 []
    |  MapPartitionsRDD[2] at flatMap at twitterFollowerCount.scala:26 []
    |  s3://fall2020-lspdp-spark/twInput MapPartitionsRDD[1] at textFile at
twitterFollowerCount.scala:25 []
    |  s3://fall2020-lspdp-spark/twInput HadoopRDD[0] at textFile at
twitterFollowerCount.scala:25 []
```

Run 2:

```
20/09/24 16:50:37 INFO root: The value of count(20) ShuffledRDD[4] at reduceByKey at
twitterFollowerCount.scala:28 []
 +-(20) MapPartitionsRDD[3] at map at twitterFollowerCount.scala:27 []
    |  MapPartitionsRDD[2] at flatMap at twitterFollowerCount.scala:26 []
    |  s3://fall2020-lspdp-spark/twInput MapPartitionsRDD[1] at textFile at
twitterFollowerCount.scala:25 []
```

```
| s3://fall2020-lspdp-spark/twInput HadoopRDD[0] at textFile at twitterFollowerCount.scala:25 []
```

3. RUNNING TIME MEASUREMENTS:

3. a) Running time of program for total of 4 runs:

Map Reduce running time (Run 1): Start: 22:40:27
End: 22:41:21
Time: 54 Seconds.

Map Reduce running time (Run 2): Start: 19:15:52
End: 19:17:00
Time: 1 minute 11 seconds.

Spark running time (Run 1): Start: 16:21:58
End: 16:22:47
Time: 49 Seconds

Spark running time (Run 2): Start: 16:50:21
End: 16:51:10
Time: 1 minute 1 seconds.

3. *b) Amount of data transferred:*

To mappers: 1319448357 => 1.319 Gb
From mappers to reducers: 961483442 => 961.4 Mb
From reducers to output: 67641452 => 67.64 Mb

3. c) Why or why not map reduce program is expected to have a good speedup!

The map reduce program is expected to have a good speedup. Closely looking at the log files reveal the job split as : org.apache.hadoop.mapreduce.JobSubmitter (main): number of splits:20. Each of these jobs are then split into 20 map tasks as specified by log (Launched map tasks=20). The output from these map tasks is then provided to 9 reducer tasks which results in the number of output of file as being 9.

There is a very small portion of the code that is inherently sequential and most of the program has a good amount of parallelism. If we break the program into small parts, first is the input where the large file ~ 1.3 gb is already split into smaller chunks, further these smaller chunks are given to multiple map tasks which process these files without any dependency on each other. The only bottleneck is where all these map jobs need to be completed so that their output can be sent to reducer, apart from that there is nothing much that hampers the parallel execution.

There could be few hardware limitations which can prevent the program from achieving a good amount of parallelization including the number of containers (If they are limited, not enough number of mappers and reducers can run) , not enough memory within these containers etc.

3. d) Syslog and stderr logs:

Map reduce Run 1:
https://github.com/2020-F-CS6240/homework-1-mapreduce-rahulpandeycs/blob/master/logs/syslog_6_m5XLarge_run2.txt

Map reduce Run 2:
https://github.com/2020-F-CS6240/homework-1-mapreduce-rahulpandeycs/blob/master/logs/syslog_6_m5xlarge_run1

Spark job run 1:
https://github.com/2020-F-CS6240/homework-1-spark-rahulpandeycs/blob/master/logs/stderr_run1_5core_1master_m5large.txt

Spark job run 2:
https://github.com/2020-F-CS6240/homework-1-spark-rahulpandeycs/blob/master/logs/stderr_run2_5core_1master_m5Large.txt

3. e) Map Reduce and Spark job output folders

Map reduce: run 1:
https://github.com/2020-F-CS6240/homework-1-mapreduce-rahulpandeycs/tree/master/outputTwitterAws1

Map reduce: run 2:
https://github.com/2020-F-CS6240/homework-1-mapreduce-rahulpandeycs/tree/master/outputTwitterAws2

Spark Run1:
https://github.com/2020-F-CS6240/homework-1-spark-rahulpandeycs/tree/master/twOutput

Spark Run 2:
https://github.com/2020-F-CS6240/homework-1-spark-rahulpandeycs/tree/master/twOutput2