CS6240

Rachna Reddy Melacheruvu

HW 1

https://github.ccs.neu.edu/melacheruvur/cs6240/tree/master/Assignment1/MapReduce/MR-Demo

https://github.ccs.neu.edu/melacheruvur/cs6240/tree/master/Assignment1/Spark/Spark-Demo

GitHub URLs:

MapReduce-

1. **Project:**
   https://github.ccs.neu.edu/melacheruvur/cs6240/tree/master/Assignment1/MapReduce/MR-Demo

2. **Logs:**
   Run 1:-
   https://github.ccs.neu.edu/melacheruvur/cs6240/blob/master/Assignment1/MapReduce/MR-Demo/aws-logs/run1/syslog-MR-1.txt
   Run 2:-
   https://github.ccs.neu.edu/melacheruvur/cs6240/blob/master/Assignment1/MapReduce/MR-Demo/aws-logs/run2/syslog-MR-2.txt

3. **Output:**
   Run 1:-
   https://github.ccs.neu.edu/melacheruvur/cs6240/tree/master/Assignment1/MapReduce/MR-Demo/outputMR1
   Run 2:-
   https://github.ccs.neu.edu/melacheruvur/cs6240/tree/master/Assignment1/MapReduce/MR-Demo/outputMR2

Spark:

1. **Project:**
   https://github.ccs.neu.edu/melacheruvur/cs6240/tree/master/Assignment1/Spark/Spark-Demo

2. **Logs:**
   Run 1:-
   https://github.ccs.neu.edu/melacheruvur/cs6240/blob/master/Assignment1/Spark/Spark-Demo/aws-logs/run1/stderr-spark-1.txt
   Run 2:-
   https://github.ccs.neu.edu/melacheruvur/cs6240/blob/master/Assignment1/Spark/Spark-Demo/aws-logs/run2/stderr-Spark2.txt

3. **Output**

Run 1:-
https://github.ccs.neu.edu/melacheruvur/cs6240/tree/master/Assignment1/Spark/Spark-Demo/outputSpark1
Run 2:-
https://github.ccs.neu.edu/melacheruvur/cs6240/tree/master/Assignment1/Spark/Spark-Demo/outputSpark2

Map-Reduce Implementation

Pseudo-code

```
Mapper {
        map()
        {
            String s = covert value to string
            String[] followers = s.split(“,”)      // split string on delimited as “,”
            String user = followers[1]           //extract the 2nd element from above result
            write a key-value pair as (user,1) to context
        }
}
Reducer {
        reduce(key,values)
        {
            sum = 0
            for each value of values for the key
            {
                sum = sum + value
            }
            Write key and sum to context

        }
}
```

Main – Idea
The program read line by line of the input file given as argument.
The idea is to keep track of number of followers each user has, so since the string is in format “user1, user2” which means, user1 follows user2.
The map function of Mapper class parses each line and splits the string based on “,” and extracts the 2nd element of the output array as that’s the user id we need to keep track.
It initializes each user’s followers as 1 and then the combiner groups by the user id and aggregates the followers in that particular input chunk.
The reducer function of Reducer class computes the final number of followers for each user considering the entire input file’s context.

Scala Implementation

Pseudo-code

```
InputFile.map(line => line.split(“,”)(1))    //extract the 2nd element of the split string based on “,”
                                             // for each line in input file.
          .map(user => (user,1))             // a pair of RDDs with key as user and value as 1
          .reduce(group by key and sum the values)
Output (user, number_of_followers) to the outputfile.
```

Usage of toDebugString

```
 2019-01-25 21:28:02 INFO  FileInputFormat:249 - Total input paths to process : 1
2019-01-25 21:28:03 INFO  root:29 - (40) ShuffledRDD[4] at reduceByKey at UserFollower.scala:28 []
 +-(40) MapPartitionsRDD[3] at map at UserFollower.scala:27 []
    |  MapPartitionsRDD[2] at map at UserFollower.scala:26 []
    |  input MapPartitionsRDD[1] at textFile at UserFollower.scala:25 []
    |  input HadoopRDD[0] at textFile at UserFollower.scala:25 []
```

<u>Running Time Measurements</u>

Map-Reduce:

	Run 1	***Run 2***
Start Time	19:43:26	20:13:49
End Time	19:45:20	20:15:59
Time Taken (in sec)	114 sec	130 sec
Data to Mapper	85331845 (records)	85331845 (records)
Data from Mapper to Reducer	15362582 (records)	15362582 (records)
Data from Reducer to Output	6626985 (records)	6626985 (records)

Run 1:
Number of Map tasks = 20
Number of reduce tasks = 12

Run 2:
Number of Map tasks = 21
Number of reduce tasks = 12

Average time of both runs = 122 sec
Although tasks are parallelized, since data communication between Mapper and Reducer takes some time, the positive effects of parallelization are negated. This can limit the speed-up, even if tasks are split into more smaller tasks and run on many more nodes.

Spark

	Run 1	***Run 2***
Start Time	20:36:25	21:09:23
End Time	20:37:40	21:10:37
Time Taken (in sec)	75 sec	74 sec

Average time of both runs = 74.5 sec