Homework1-Section1-Yexin Wang

Map-Reduce Implementation (30 points total)

Pseudo-code

```
map(Object o, Text line)
  // the line contains "user1, user2"
  // get user2 which means get the user who is being followed
  user u = line.getUser2
  emit(u, 1)

combine(user u, [c1, c2, c3])
  // combine is the same as reduce except its done in mapper

reduce(user u, [c1, c2, c3, ...])
  // each c is a partial count
  total = 0
  for each c in list
      total += c
  emit(u, total)
```

Main idea

My programs read the input line by line. The map functions parses a line to extract the user who is being followed. For each user extracted, it outputs the user with value 1(user => (user, 1)). These (user, number) pairs are grouped by the user and then the reduce function computes a sum for each group.

Spark Scala Implementation (30 points total)

Pseudo-code

```
val textFile = sc.textFile(args(0))
// filter the map to get the user who is being followed
val filteredMap = textFile.flatMap(line =>
line.split(",")).zipWithIndex.collect {
  case (x, i) if i % 2 != 0 => x
}
val counts = filteredMap.map(user => (user, 1)).reduceByKey(_ + _)
logger.info(counts.toDebugString)
counts.saveAsTextFile(args(1))
```

toDebugString

```
19/09/22 22:42:48 INFO root: (20) ShuffledRDD[7] at reduceByKey at
FollowerCount.scala:31 []
 +-(20) MapPartitionsRDD[6] at map at FollowerCount.scala:31 []
    |   MapPartitionsRDD[5] at collect at FollowerCount.scala:27 []
    |   MapPartitionsRDD[4] at collect at FollowerCount.scala:27 []
    |   ZippedWithIndexRDD[3] at zipWithIndex at FollowerCount.scala:27 []
    |   MapPartitionsRDD[2] at flatMap at FollowerCount.scala:27 []
    |   s3://elricsparkbucket/input MapPartitionsRDD[1] at textFile at
FollowerCount.scala:25 []
    |   s3://elricsparkbucket/input HadoopRDD[0] at textFile at
FollowerCount.scala:25 []
```

Running Time Measurements(12 points total)

Running time

MapReduce:(using 6 m4.xlarge)

1. 2 minutes 11 seconds
2. 1 minute 24 seconds

Spark:(using 6 m4.xlarge)

1. 1 minute 11 seconds

2. 1 minute 7 seconds

Amount of data transferred

- To the Mappers : 1319473741bytes
- Mappers to Reducers: 961483442 bytes
- Reducers to output : 67641452 bytes

About Speed up

I think my MapReduce program is expected to have good speedup. There are 21 tasks in map stage and 20 reduce tasks in reduce stage. These two stages are parallelizable. As a result, I don't think there is an inherently sequential part in my program.

links

log file:

- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-1/tree/master/MR-Demo/first-run/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-1/tree/master/MR-Demo/second-run/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-1/tree/master/Spark-Demo/first-run/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-1/tree/master/Spark-Demo/second-run/log

output file:

- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-1/tree/master/MR-Demo/first-run/output
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-1/tree/master/MR-Demo/second-run/output
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-1/tree/master/Spark-Demo/first-run/output
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-1/tree/master/Spark-Demo/second-run/output