Homework3-Section1-Yexin Wang

https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3

Combining in Spark (20 points total)

RDDG

- Pseudo-code

```
    val textFile = sc.textFile(args(0))
    val counts = textFile.map(line => line.split(",")(1))
      // map to a pairRDD
      .map(userId => (userId, 1))
      // alter the pair RDD so that the same key are gathered together
      .groupByKey()
      .mapValues(count => count.sum)
      .sortBy(_._2, ascending = false)
    // Output the counting file
    counts.saveAsTextFile(args(1))
```

- toDebugString

```
|   ShuffledRDD[9] at sortBy at FollowerRDDG.scala:25 []
 +-(40) MapPartitionsRDD[6] at sortBy at FollowerRDDG.scala:25 []
    |   MapPartitionsRDD[5] at mapValues at FollowerRDDG.scala:24 []
    |   ShuffledRDD[4] at groupByKey at FollowerRDDG.scala:23 []
    +-(40) MapPartitionsRDD[3] at map at FollowerRDDG.scala:21 []
       |   MapPartitionsRDD[2] at map at FollowerRDDG.scala:19 []
       |   input/edges.csv MapPartitionsRDD[1] at textFile at
FollowerRDDG.scala:18 []
       |   input/edges.csv HadoopRDD[0] at textFile at FollowerRDDG.scala:18 []
```

conclusion: shuffling before aggregation

RDD-R

- Pseudo-code

```
val textFile = sc.textFile(args(0))
    val counts = textFile.map(line => line.split(",")(1))
      // map to a pairRDD
      .map(userId => (userId, 1))
      .reduceByKey(_ + _)
      .sortBy(_._2, ascending = false)
    // Output the counting file
    counts.saveAsTextFile(args(1))
```

- toDebugString

```
|   ShuffledRDD[8] at sortBy at FollowerRDDR.scala:23 []
 +-(40) MapPartitionsRDD[5] at sortBy at FollowerRDDR.scala:23 []
    |   ShuffledRDD[4] at reduceByKey at FollowerRDDR.scala:22 []
    +-(40) MapPartitionsRDD[3] at map at FollowerRDDR.scala:21 []
       |   MapPartitionsRDD[2] at map at FollowerRDDR.scala:19 []
       |   input/edges.csv MapPartitionsRDD[1] at textFile at
FollowerRDDR.scala:18 []
       |   input/edges.csv HadoopRDD[0] at textFile at FollowerRDDR.scala:18 []
```

conclusion: aggregation before shuffling

RDD-F

- Pseudo-code

```
    val textFile = sc.textFile(args(0))
    val counts = textFile.map(line => line.split(",")(1))
      // map to a pairRDD
      .map(userId => (userId, 1))
      // merge the values for each key
      .foldByKey(0)((count1, count2) => count1 + count2)
      .sortBy(_._2, ascending = false)
    // Output the counting file
```

```
    counts.saveAsTextFile(args(1))
```

- toDebugString

```
 |   ShuffledRDD[8] at sortBy at FollowerRDDF.scala:24 []
 +-(40) MapPartitionsRDD[5] at sortBy at FollowerRDDF.scala:24 []
    |   ShuffledRDD[4] at foldByKey at FollowerRDDF.scala:23 []
    +-(40) MapPartitionsRDD[3] at map at FollowerRDDF.scala:21 []
       |   MapPartitionsRDD[2] at map at FollowerRDDF.scala:19 []
       |   input/edges.csv MapPartitionsRDD[1] at textFile at
FollowerRDDF.scala:18 []
       |   input/edges.csv HadoopRDD[0] at textFile at FollowerRDDF.scala:18 []
```

conclusion: aggregation before shuffling

RDD-A

- Pseudo-code

```
    val textFile = sc.textFile(args(0))
    val counts = textFile.map(line => line.split(",")(1))
      // map to a pairRDD
      .map(userId => (userId, 1))
      .aggregateByKey(0)(_+_, _+_)
      .sortBy(_._2, ascending = false)
    // Output the counting file
    counts.saveAsTextFile(args(1))
```

- toDebugString

```
(40) MapPartitionsRDD[9] at sortBy at FollowerRDDA.scala:23 []
 |   ShuffledRDD[8] at sortBy at FollowerRDDA.scala:23 []
 +-(40) MapPartitionsRDD[5] at sortBy at FollowerRDDA.scala:23 []
    |   ShuffledRDD[4] at aggregateByKey at FollowerRDDA.scala:22 []
    +-(40) MapPartitionsRDD[3] at map at FollowerRDDA.scala:21 []
       |   MapPartitionsRDD[2] at map at FollowerRDDA.scala:19 []
```

```
       |   input/edges.csv MapPartitionsRDD[1] at textFile at
FollowerRDDA.scala:18 []
       |   input/edges.csv HadoopRDD[0] at textFile at FollowerRDDA.scala:18 []
```

conclusion: aggregation before shuffling

DSET

- Pseudo-code

```
val dataSet: RDD[String] = sparkSession.sparkContext.textFile(args(0))

val counts = dataSet.map(line => (line.split(",")(1), 1))
      .toDS()
      .groupBy("_1")
      .sum("_2")
      .orderBy(desc("sum(_2)"))

    counts.write.csv(args(1))
```

- explain(extended=true)

```
== Parsed Logical Plan ==
'Sort ['sum(_2) DESC NULLS LAST], true
+- AnalysisBarrier
      +- Aggregate [_1#3], [_1#3, sum(cast(_2#4 as bigint)) AS sum(_2)#9L]
         +- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType, fromString,
assertnotnull(assertnotnull(input[0, scala.Tuple2, true]))._1, true, false) AS
_1#3, assertnotnull(assertnotnull(input[0, scala.Tuple2, true]))._2 AS _2#4]
            +- ExternalRDD [obj#2]

== Analyzed Logical Plan ==
_1: string, sum(_2): bigint
Sort [sum(_2)#9L DESC NULLS LAST], true
+- Aggregate [_1#3], [_1#3, sum(cast(_2#4 as bigint)) AS sum(_2)#9L]
   +- SerializeFromObject [staticinvoke(class
```

```
org.apache.spark.unsafe.types.UTF8String, StringType, fromString,
assertnotnull(assertnotnull(input[0, scala.Tuple2, true]))._1, true, false) AS
_1#3, assertnotnull(assertnotnull(input[0, scala.Tuple2, true]))._2 AS _2#4]
      +- ExternalRDD [obj#2]

== Optimized Logical Plan ==
Sort [sum(_2)#9L DESC NULLS LAST], true
+- Aggregate [_1#3], [_1#3, sum(cast(_2#4 as bigint)) AS sum(_2)#9L]
   +- SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType, fromString,
assertnotnull(input[0, scala.Tuple2, true])._1, true, false) AS _1#3,
assertnotnull(input[0, scala.Tuple2, true])._2 AS _2#4]
      +- ExternalRDD [obj#2]

== Physical Plan ==
*(3) Sort [sum(_2)#9L DESC NULLS LAST], true, 0
+- Exchange rangepartitioning(sum(_2)#9L DESC NULLS LAST, 200)
   +- *(2) HashAggregate(keys=[_1#3], functions=[sum(cast(_2#4 as bigint))],
output=[_1#3, sum(_2)#9L])
      +- Exchange hashpartitioning(_1#3, 200)
         +- *(1) HashAggregate(keys=[_1#3], functions=[partial_sum(cast(_2#4 as
bigint))], output=[_1#3, sum#16L])
            +- *(1) SerializeFromObject [staticinvoke(class
org.apache.spark.unsafe.types.UTF8String, StringType, fromString,
assertnotnull(input[0, scala.Tuple2, true])._1, true, false) AS _1#3,
assertnotnull(input[0, scala.Tuple2, true])._2 AS _2#4]
               +- Scan ExternalRDDScan[obj#2]
```

conclusion: shuffling before aggregation

Join Implementation (48 points total)

RS_R

- Pseudo-code

```
val user_first = textFile
```

```
      // map to a tuple
      .map(line => (line.split(",")(0), line.split(",")(1)))
      // convert string to int
      .filter(line => line._1.toInt <= MAX && line._2.toInt <= MAX)

    // use id2 as the key
    val user_second = textFile
      .map(line => (line.split(",")(1), line.split(",")(0)))
      .filter(line => line._1.toInt <= MAX && line._2.toInt <= MAX)

    // we compute a path2, tuple join a tuple will result in a format like(41,
(126,140))
    // we have to filer the tuple like (96,(41,41))
    val path2 = user_second.join(user_first).filter(line => line._2._1 !=
line._2._2)
      // use null since we don't care value
      .map(line => ((line._2._2, line._2._1), null))

    val users = user_first.map(line => ((line._1, line._2), null))
    val triangle = path2.join(users)
    // divide by 3 since we calculate each triangle third times
    val triangleCount = triangle.count() / 3
```

RS_D

- Pseudo-code

```
 val user = textFile
      .map(line => (line.split(",")(0), line.split(",")(1)))
      .toDF("id1", "id2").filter($"id1" <= MAX && $"id2" <= MAX)

    /* +---+---+---+---+
       |id1|id2|id1|id2|
       +---+---+---+---+
       |467|534| 41|467|
       |675| 41|534|675|
       |675|534| 41|675|*/
```

```
    val path2 = user.as("first").join(user.as("second"))
      .filter($"first.id1" === $"second.id2" && $"first.id2" =!= $"second.id1")
      // rename
      .toDF("_a", "id3", "id1", "_b").drop("_a", "_b")
    val triangle = path2.as("path2").join(user.as("user"))
      .filter($"path2.id1" === $"user.id2" && $"path2.id3" === $"user.id1")
    val count = triangle.count() / 3
```

Rep_R

- Pseudo-code

```
val edge = textFile
      .map(line => (line.split(",")(0), line.split(",")(1)))
      .filter(line => line._1.toInt <= MAX && line._2.toInt <= MAX)

    // create a tuple (id2, (id1, id2))
    val second = edge.map(line => (line._2, line))

    // create a new hash set, any iterable should work. just like linked list I
used in
    // map reduce rep join
    val hashSet = mutable.HashSet.empty[String]
    // function, add an element to the set
    val addToSet = (set: mutable.HashSet[String], element: String) => set +=
element
    // function, merge two sets
    val mergeSet = (set1: mutable.HashSet[String], set2:
mutable.HashSet[String]) => set1 ++= set2
    // broadcast, https://learning.oreilly.com/library/view/high-performance-
spark/9781491943199/ch04.html
    val broadcastMap = sc.broadcast(edge
      // really handy, simplify the process to create a iterable as value
      .aggregateByKey(hashSet)(addToSet, mergeSet)
      // now we have a map, id -> set(user_ids which id is following)
      .collectAsMap())

    val path2: RDD[(String, String)] = second.flatMap {
```

```
      case (id2, record) => broadcastMap.value.get(id2) match {
        // seq.empty has type but does not hold value
        case None => Seq.empty[(String, (String, String))]
        // Some indicate the option is valid, get the hash set
        // assemble a path2 here, (id1, id3)
        case Some(id) => id.map(id3 => (record._1, id3))
      }
    }.map(line => (line._1, line._2.toString))

    val triangle: RDD[(String, io.Serializable)] = path2.flatMap {
      // try to get the id3 set and examine if id1 in it
      case (id1, id3) => broadcastMap.value.get(id3) match {
        case None => Seq.empty[(String, (String, String))]
          // get each element in the id3 hash set, check if the element is id1
        case Some(id) => id.map(id_1 => (id1, id_1))
          .filter(line => line._1 == line._2)
      }
    }

    val triangleCount = sc.parallelize(
      Seq("TriangleCount: ", triangle.count() / 3))
```

Rep_D

- Pseudo-code

```
val textFile = sparkSession.sparkContext.textFile(args(0))
val user = textFile
  .map(line => (line.split(",")(0), line.split(",")(1)))
  .toDF("id1", "id2").filter($"id1" <= MAX && $"id2" <= MAX)

// force to broadcast
val bUser = broadcast(user)

/* +---+---+---+---+
   |id1|id2|id1|id2|
   +---+---+---+---+
   |467|534| 41|467|
```

```
   |675| 41|534|675|
   |675|534| 41|675|*/

val path2 = user.as("first").join(bUser.as("second"))
  .filter($"first.id1" === $"second.id2" && $"first.id2" =!= $"second.id1")
  // rename
  .toDF("_a", "id3", "id1", "_b").drop("_a", "_b")

val triangle = path2.as("path2").join(bUser.as("user"))
  .filter($"path2.id1" === $"user.id2" && $"path2.id3" === $"user.id1")

val count = triangle.count() / 3

val result = sparkSession.sparkContext.parallelize(Seq("TriangleCount: ",
count))

// Output the counting file
result.saveAsTextFile(args(1))
println("The count of triangles: " + count)
path2.explain()
triangle.explain()
```

Result Table

Configuration	Small Cluster Result	Large Cluster Result
RS-R, MAX = 15000	Running time: 8 min. Triangle count: 1, 096, 152	Running time: 4.5 min. Triangle count: 1, 096, 152
RS-D, MAX = 50000	Running time: 16.5 min. Triangle count: 12, 029, 907	Running time: 10 min. Triangle count: 12, 029, 907
Rep-R, MAX = 15000	Running time: 29 min. Triangle count: 1, 096, 152	Running time: 29 min. Triangle count: 1, 096, 152
Rep-D, MAX = 50000	Running time: 3 min. Triangle count: 12, 029, 907	Running time: 2 min 30s. Triangle count: 12, 029, 907

(I am using markdown so it is hard to generate table, so I use word to draw the table and screenshot it. Sorry for the inconvenience.)

links

log file:

- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/RS_R%206%20m4.large/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/RS_R%2011%20m4.large/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/RS_D%206%20m4.large/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/RS_D%2011%20m4.large/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/Rep_R%206%20m4.large/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/Rep_R%2011%20m4.large/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/Rep_D%206%20m4.large/log
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/Rep_D%2011%20m4.large/log

output file:

- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/RS_R%206%20m4.large/output
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/RS_R%2011%20m4.large/output
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/RS_D%206%20m4.large/output
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/RS_D%2011%20m4.large/output
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/Rep_R%206%20m4.large/output
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/Rep_R%2011%20m4.large/output
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/Rep_D%206%20m4.large/output
- https://github.ccs.neu.edu/cs6240-f19/wangyexin-Assignment-3/tree/master/Spark-Demo/Rep_D%2011%20m4.large/output