CS455 – Lab Session 11

NAMAN SHAH

04/07/2017

Agenda

- Quiz-8 review
- Spark Example

Quiz-8

1. Memory residency of data is one of the key enabling concepts in Spark. [True/False]

2. RDD Transformation operators in Spark do not necessarily result in an RDD. [True/False]

3. Actions on an RDD must result in an RDD. [True/False]

4. Spark consults the lineage graph for an RDD as part of the sequence of operations leading up to loading portions of the data in memory. [True/False]
Quiz-8
5. The primary objective of lazy evaluations of RDDs is to avoid repeated passes over the same underlying files. [True/False]

6. Once persisted, an RDD is memory resident on the machine where the driver program is executing. [True/False]

Quiz-8
7. RDDs are recomputed (by default) every time you run an action on them. [True/False]

8. Transformation operations on RDDs can involve at most 2 RDDs. [True/False]

Quiz-8
9. Actions on an RDD do not necessarily have data locality. [True/False]

10. Pair RDDs expose partitions that allow you to act on each key in parallel or regroup data across the network. [True/False]

Spark Example (Q1 in HW3-PC)

```java
public static void main(String[] args) throws IOException {
    SparkConf sparkConf = new SparkConf()
        .setAppName("cs455-hw3-spark-example");
    JavaSparkContext jsc = new JavaSparkContext(
        sparkConf);
    Configuration hadoopConfiguration = jsc.hadoopConfiguration();
    JavaRDD<String> miniDataset = jsc.textFile("hdfs://…");
    JavaRDD<String> filterForQ1 = miniDataset.filter((String line) -> {
        String summaryLevel = line.substring(10, 13);
        String segmentNo = line.substring(24, 28);
        return summaryLevel.equals("100") &&
            segmentNo.equals("0002");});
    filterForQ1.cache();
```
Spark Example (Q1 in HW3-PC) cont...

```java
JavaPairRDD<String, Tuple2<Long, Long>> combinedRDD = filterForQ1.mapToPair((String line) -> {
    String state = line.substring(8, 10);
    long owned = Long.parseLong(line.substring(1803, 1812));
    long rented = Long.parseLong(line.substring(1812, 1821));
    return new Tuple2<>(state, new Tuple2<>(owned, rented));
}).reduceByKey((Tuple2<Long, Long> t1, Tuple2<Long, Long> t2) -> new Tuple2<>(t1._1 + t2._1, t1._2 + t2._2));
```

```java
JavaPairRDD<String, Tuple2<Double, Double>> answerRDD = combinedRDD.mapValues((Tuple2<Long, Long> t1) -> new Tuple2<>(t1._1 * 100d / (t1._1 + t1._2), t1._2 * 100d / (t1._1 + t1._2)));
List<Tuple2<String, Tuple2<Double, Double>>> answer = answerRDD.collect();
for (Tuple2<String, Tuple2<Double, Double>> answer1 : answer) {
    System.out.println(answer1._1 + " OWNED: " + answer1._2._1 + " RENTED: " + answer1._2._2);
}
```

QUESTIONS?