Questions from the paper "RDD: A Fault-Tolerant Abstraction for In-Memory Cluster Computing "

Paper can be downloaded from:
https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf

1. Introduction

a. Existing cluster computing frameworks lack abstractions for leveraging distributed shared memory. This makes them inefficient for two classes of applications. Name these two classes and explain them briefly.

b. What are some of the design considerations for RDDs? How is the problem of efficiently providing fault tolerance solved in case of RDDs?

2. RDDs

2.1 RDD Abstraction

a. What are the two deterministic ways in which RDDs can be created?

b. Which properties of RDDs can users control?

2.2 Spark Programming Interface

a. Understand actions in Spark. What do they return? Why do you think Spark computes RDDs lazily the first time they are used in an action?

b. Look at the example 2.2.1 and understand the code. What method can you call on an RDD so that it stays in memory even after an action command has been issued on it?

2.3 Advantages of the RDD model

a. What are the advantages of RDD model over traditional DSM model

3. Spark Programming Interface

-- No written answers required for this section, but be sure to understand the concepts and examples carefully –
-- Understand the transformations and actions presented in Table 2 carefully –
-- You can also consult the Spark programming guide:
http://spark.apache.org/docs/latest/programming-guide.html

4. Representing RDDs

a. What are the 5 pieces of information that the RDD common interface exposes? Explain them briefly.

b. What is the difference between *narrow* and *wide* dependencies for RDDs?

c. Which type of dependency needs data from multiple nodes? Which type of dependency makes recovery from failure easier?

d. What would the **partitions()** operation would return for following cases:

- HDFS files
- Instance of MappedRDD object
- An RDD created by union method
- An RDD created by sample method
- An RDD created by join method

5. Implementation

5.1 Job Scheduling

a. The scheduler tries to build stages by combining transformation that have what type of dependencies?

b. What do the boundary of the stages represent?

c. How does Spark use the concept of data locality?

5.3 Memory Management

a. What 3 options does Java provide for storage of persistent RDDs? Which one provides fastest performance and which one is best suited for large RDDs

6. Evaluation

-- Read the sections and performance improvement metrics –
-- No written answers needed –

7. Discussion

-- Read the sections and understand how various programming models can be expressed in Spark --
-- No written answered needed --