1 Section C - Concepts in Apache Spark and Distributed Computing

Please answer in full sentence(s) (max 3), per question, as appropriate. Use the handout to help.

1.1 C.1 Why do we use the Map-Reduce programming model

Map-Reduce programming model is able to process and generate big data sets with a parallel, distributed algorithm on a cluster, and it is executed on the server where structured, semi-structured, or unstructured data is stored with for redundancy and fault tolerance. Besides, there are open-source implantation of it with APIs in many programming languages with different level optimization for usage.

1.2 C.2 Anna's trying to understand her Spark code

The single words in the output list proves that defined function has worked although the print content is not shown. The unseen reason is that flatMap method just returns a new RDD in the partitions, and during the process, the defined function is running in the server or cluster instead of conducted and outputting in the local machine.

1.3 Calling .collect() on a large dataset can cause my driver application to run out of memory, why?

Because collect() method fetches the entire RDD to the local single machine, thus when the processed data is in large size, the driver application will run out of the memory or even crash.

1.4 C.4 Are partitions mutable or immutable? Why is this advantageous?

Partitions are immutable because the RDDs are stored inside it and RDDs are not mutable for concurrency and simplicity. A partition is an atomic chunk of data (logical division of data) stored on a node in the cluster. The immutable partitions are basic units of parallelism in Apache Spark and allows to distribute the work across the cluster, divide the task into smaller parts, and reduce memory requirements for each node.

1.5 C.5 In what sense are RDDs 'resilient'? How is this achieved?

Resilient here means the fault tolerance to the data set and compute workers, and leads the data to a fault-tolerant collection of elements that can be operated on in parallel.

It is achieved by using a lineage graph, disk IO and lazy evaluation. The RDDs recourse manager controls over scheduling and resilient processing to reduce the time of writing and reading files to disk. The lazy evaluation which leads to that transformations, like map function, are only computed when an action, like reduce operation, requires a result to be returned to the driver program. Besides, the initial design of the system allows RDDs automatically recover from node failures.

2 Section D - Essay Questions

2.1 A colleague has mentioned her Spark application has poor performance, what is your advice?

(List 4 clear recommendations, answer in full sentences. I suggest 1 or 2 sentences for each recommendation.)

1. Use Java serialization or Kryo serialization to serializes objects, design the data structure into arrays of objects, or primitive types (fastutil library can be used for this), instead of the standard Java or Scala collection classes (e.g. HashMap).

2. Use persist() or cache methods to have the RDDs, which will be operated in many times, into memory or disk, reduce the number of RDDs by creating a RDD containing more data, and avoid to create RDDs of the same file for many times.

3. Try to avoid shuffles by reducing usage of methods reduceByKey, join, distinct, repartition etc., and try to use map-side combiner to replace shuffle operations.

4. Upgrade the hardware like CPU, memory storage, hard disk to SSD, and Internet connection etc.