Chapter 14: Hello Apache Spark

Exercises :

Explore available transformations and actions of Apache Spark RDD APIs. For example, transformations: foreach, reduce and zipWithIndex and actions: collect, action, take, and first. Use them in your code. You can use them without any barrier. You have all the required knowledge of Scala and Spark to do that. So trust yourself and do that.

Answers :

Apache Spark

Spark is a unified analytics engine for large-scale data processing. It provides high-level APIs in Scala, Java, Python, and R, and an optimized engine that supports general computation graphs for data analysis. It also supports a rich set of higher-level tools including Spark SQL for SQL and DataFrames, pandas API on Spark for pandas workloads, MLlib for machine learning, GraphX for graph processing, and Structured Streaming for stream processing.

At a high level, every Spark application consists of a driver program that runs the user's main function and executes various parallel operations on a cluster. The main abstraction Spark provides is a resilient distributed dataset (RDD), which is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel. RDDs are created by starting with a file in the Hadoop file system (or any other Hadoop-supported file system), or an existing Scala collection in the driver program, and transforming it. Users may also ask Spark to persist an RDD in memory, allowing it to be reused efficiently across parallel operations. Finally, RDDs automatically recover from node failures.

A second abstraction in Spark is shared variables that can be used in parallel operations. By default, when Spark runs a function in parallel as a set of tasks on different nodes, it ships a copy of each variable used in the function to each task. Sometimes, a variable needs to be shared across tasks, or between tasks and the driver program. Spark supports two types of shared variables: broadcast variables, which can be used to cache a value in memory on all nodes, and accumulators, which are variables that are only "added" to, such as counters and sums.

RDD Transformations are Spark operations when executed on RDD, it results in a single or multiple new RDD's. Since RDD are immutable in nature, transformations always create new RDD without updating an existing one hence, this creates an RDD lineage.

RDD Transformations are Lazy

RDD Transformations are lazy operations meaning none of the transformations get executed until you call an action on Spark RDD. Since RDD's are immutable, any transformations on it result in a new RDD leaving the current one unchanged.

Narrow Transformation

Narrow transformations are the result of map() and filter() functions and these compute data that live on a single partition meaning there will not be any data movement between partitions to execute narrow transformations

Functions such as `map()`, `mapPartition()`, `flatMap()`, `filter()`, `union()` are some examples of narrow transformation

Wider Transformation

Wider transformations are the result of groupByKey() and reduceByKey() functions and these compute data that live on many partitions meaning there will be data movements between partitions to execute wider transformations. Since these shuffles the data, they also called shuffle transformations.
Functions such as `groupByKey()`, `aggregateByKey()`, `aggregate()`, `join()`, `repartition()`are some examples of a wider transformations.