APACHE SPARCK

Apache Spark is a “computational engine” that is responsible for scheduling, distributing, and mon‐itoring applications consisting of many computational tasks across many worker machines, or a computing cluster

Apache Spark is a fast and general purpose engine for large-scale data processing. Under the hood, it works on a cluster of computers

There are two things.

1. A cluster computing engine.
2. A set of libraries, APIs, and DSLs.

Apache Spark is a distributed processing engine, but it doesn't come with an inbuilt cluster resource manager and a distributed storage system. You have to plugin a cluster manager and a storage system of your choice. There are multiple alternatives. You can use Apache YARN, Mesos, and Kubernetes as a cluster manager for Apache Spark.

Similarly, for the storage system, you can use HDFS, Amazon S3, Google Cloud storage, Cassandra File system and much more

## Spark Core APIs

Spark core consists of two APIs.

1. Structured API
2. Unstructured API

The Structured APIs consists of data frames and data sets. They are designed and optimized to work with structured data.   
The Unstructured APIs are the lower level APIs including RDDs, Accumulators and Broadcast variables. These core APIs are available in Scala, Python, Java, and R. We will learn more about these APIs as we progress with the tutorial.   
Outside the Spark Core, we have four different set of libraries and packages.

1. Spark SQL - Allows you to use SQL queries for structured data processing.
2. Spark Streaming - Helps you to consume and process continuous data streams.
3. MLlib - A machine learning library that delivers high-quality algorithms.
4. GraphX - Comes with a library of typical graph algorithms.

RDD Basics

An RDD in Spark is simply an immutable distributed collection of objects. Each RDD is split into multiple partitions, which may be computed on different nodes of the cluster.

Spark RDD is a resilient, partitioned, distributed and immutable collection of data.

Let's quickly review this description.

1. Collection of data - This one is the most basic thing. They hold data and appears to be a Scala Collection.
2. Resilient - That means, they can recover from a failure. RDDs are fault tolerant.
3. Partitioned - Spark breaks the RDD into smaller chunks of data. These pieces are called partitions.
4. Distributed - Instead of keeping these Partitions on a single machine, Spark spreads them across the cluster. So, they are a distributed collection of data.
5. Immutable - Once defined, you can't change them. So, Spark RDD is a read-only data structure.

You can create an RDD using two methods.

1. Load some data from a source.
2. Create an RDD by transforming another RDD.

RDD OPERATIONS

1. Transformations: Transformations are operations on RDDs that return a new RDD, such asmap() and filter(). Transformations on RDDs are lazily evaluated, meaning thatSpark will not begin to execute until it sees an action.

2. Actions are operations that return a result to the driver pro‐gram or write it to storage, and kick off a computation, such as count() and first().The Actions are mainly performed to send results back to the driver, and hence they produce a non-distributed dataset.

In Brief:

1. RDDs – RDDs are the core of Spark. They represent a partitioned and distributed data set.
2. Transformations and Actions – We can perform transformations and actions over the RDDs.
3. Spark Job – An action on an RDD triggers a job.
4. Stages – Spark breaks the job into stages.Shuffle and Sort – A shuffle activity is a reason to break the job into two stages.
5. Shuffle and Sort – A shuffle activity is a reason to break the job into two stages.
6. Tasks – Each stage is executed in parallel tasks. The number of parallel tasks is directly dependent on the number of partitions.
7. Executors – Apart from the tasks, the number of available executors is also a constraint on the degree of parallelism.