Spark

What exactly is Spark? Why is there such a frenzy over this technology? Apache Spark is an open-source cluster computing system written in Java, Scala, Python, and R. It provides a high-level API in these languages. It supports data access from HDFS, Cassandra, HBase, Hive, and Tachyon, as well as any other Hadoop data source. Additionally, it can be run in standalone, YARN, or Mesos cluster managers.

The Spark tutorial will cover the components of the Spark ecosystem, a video tutorial on Spark, the Spark abstraction – RDD, transformation, and action in Spark RDD. The purpose of this introduction tutorial is to provide an in-depth overview of Spark, including its history, architecture, deployment approach, and RDD support in Spark.

What exactly is Spark?

Apache Spark is a multi-purpose, high-performance cluster computing system. It comes with a high-level API. Java, Scala, Python, and R are just a few examples. Apache Spark is a platform for developing and executing Spark applications. Spark is 100 times quicker than Hadoop and ten times faster than disc access.

Although Spark is designed in Scala, it supports a variety of programming languages including Scala, Java, Python, and R.

It is compatible with Hadoop and capable of processing existing Hadoop HDFS data. How is Spark compatible with Hadoop?

It is a proverb that a picture is worth a thousand words. Keeping this in mind, we've included a Spark video tutorial to help you gain a better knowledge of Apache Spark.

Apache Spark's History

Apache Spark was launched in 2009 at UC Berkeley's Research and Development Laboratory, afterwards renamed AMPLab. It was released as open source software under the BSD licence in 2010. Spark was contributed to the Apache Software Foundation in 2013 and was promoted to a top-level Apache project in 2014.

What is the purpose of Spark?

After reviewing the Apache Spark introduction, let us explore why Spark was created.

There is a need in industry for a general-purpose cluster computing tool for the following reasons:

- Hadoop MapReduce is a batch processing framework.
- Apache Storm / S4 is a stream processing framework.
- Apache Impala / Apache Tez are not capable of performing batch processing.
- Neo4j / Apache Giraph is restricted to graph processing.

As a result, there is a high need in the industry for a strong engine capable of processing data in real-time (streaming) as well as batch mode. There is a requirement for an engine capable of sub-second response time and in-memory processing.

According to the Apache Spark Definition, it is a strong open-source engine that enables real-time stream processing, interactive processing, graph processing, in-memory processing, and batch processing at a high rate of speed, with a simple interface. This establishes the distinction between Hadoop and Spark and also establishes a significant contrast between Spark and Storm.

We examined the meaning of spark, its history, and its significance in our What is Spark tutorial. Now, let us discuss spark components.

Components for Apache Spark

Apache Spark makes the promise of increased data processing speed and development ease. How does Spark accomplish this? To begin answering this question, let's discuss the Apache Spark ecosystem, which is a critical component of the Apache Spark introduction since it enables Spark to be quick and dependable. These Spark components address the challenges that arose when utilising Hadoop MapReduce.

Components of the Spark Ecosystem

Here, we'll go over each component of the Spark Ecosystem one by one.

i. Spark Core

It is the Spark kernel that serves as the execution platform for all Spark applications. It is a general-purpose platform capable of supporting a diverse range of applications.

ii. Spark SQL

On top of Spark, it enables users to run SQL/HQL queries. We can process structured and semi-structured data with Apache Spark SQL. Additionally, it includes a Hive engine that enables unmodified queries to run up to 100 times quicker on current deployments. For a more extensive examination, go to the Spark SQL Tutorial.

iii. Spark Streaming

Apache Spark Streaming enables the development of highly interactive and data-analytic applications that operate on live streaming data. The live feeds are transformed into micro-batches and executed on top of spark core. For a full examination of Apache Spark Streaming, please refer to our Spark Streaming tutorial.

iv. Spark MLlib

It is the scalable machine learning library that ensures both efficiency and algorithm quality. Apache Spark MLlib is a popular choice for Data Scientists because to its in-memory data processing capacity, which significantly enhances the efficiency of iterative algorithms.

v. Spark GraphX

Apache Spark GraphX is a graph computation engine developed on top of Spark that enables massively parallel processing of graph data.

vi. SparkR

It is a R package that provides a lightweight frontend for interacting with Apache Spark from within R. It enables data scientists to study huge datasets and conduct jobs on them interactively from the R shell. The primary goal of SparkR was to investigate various strategies for integrating the usability of R and the scalability of Spark.

For a full examination of Spark components, see the Spark Ecosystem Guide.

RDD - Resilient Distributed Dataset

In this section of the Apache Spark Tutorial, we will cover the RDD abstraction, which is a critical component of Spark.

Apache Spark's fundamental data structure is the Resilient Distributed Dataset (RDD), which is a distributed collection of elements across cluster nodes capable of performing parallel operations. While Spark RDDs are immutable, they can be transformed to generate new RDDs.

In Spark, there are three ways to construct RDDs:

- **Parallelized collections** — By using the parallelize method in the driver application, we can construct parallelized collections.
- **External datasets** - RDDs can be created by invoking the textFile function. This method reads the file's URL as a series of lines.
- **Existing RDDs** - We can construct new RDDs by performing transformation operations on existing RDDs.

Learn in detail how to create RDDs in Spark.

RDDs in Apache Spark offer two distinct sorts of operations:

- **Transformation** — Converts an existing RDD into a new one. It calls the method and returns a new dataset.
- **Initiation** – The action either returns the completed result to the driver software or writes it to an external data store.

This link will teach you how to use the RDD Transformations and Actions APIs with examples.

Spark Shell

Apache Spark includes a graphical user interface (GUI) called spark-shell. It enables Spark apps to execute simply on the system's command line. We may run/test our application code interactively using the Spark shell. Spark can read data from a variety of different sources, allowing it to access and process massive amounts of data.

Conclusion

This Spark lesson is a combination of technologies that adds value to large data and enables new Spark use cases. It provides a unified platform for developing, managing, and implementing Spark requirements for big data processing. Spark video lessons teach you all you need to know about Spark.

Along with MapReduce operations, Spark enables the execution of SQL queries and the processing of streaming data, which were shortcomings of Hadoop-1. Developers can use Spark capabilities independently or in combination with MapReduce programming techniques.