Session: 23, 24 and 25-[Apache Spark APIs for large-scale data processing]

Overview, Linking with Spark, Initializing Spark

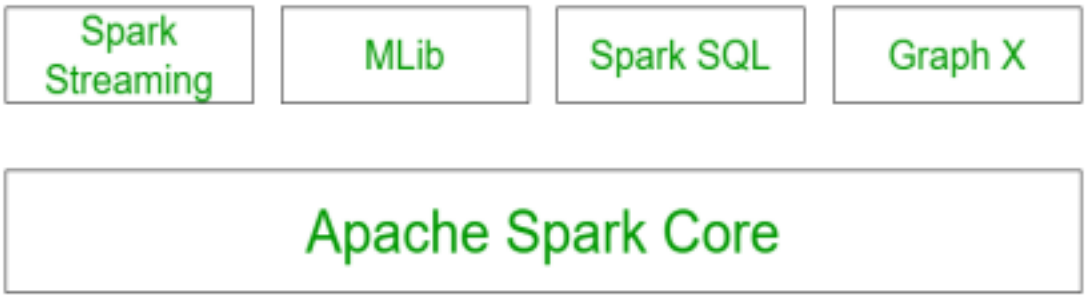
What is spark

- Spark is a cluster computing system. It provides high-level APIs in Python, Scala, and Java.
- park is written in Scala and provides API in Python, Scala, Java, and R.

Features of spark are:

1. **Multiple Language Support:** Apache Spark supports multiple languages; it provides API's written in Scala, Java, Python or R. It permits users to write down applications in several languages.
2. **Quick Speed:** The most vital feature of Apache Spark is its processing speed. It permits the application to run on a Hadoop cluster, up to one hundred times quicker in memory, and ten times quicker on disk.
3. **Runs Everywhere:** Spark will run on multiple platforms while not moving the processing speed. It will run on Hadoop, Kubernetes, Mesos, Standalone, and even within the Cloud.
4. **General Purpose:** It is powered by plethora libraries for machine learning (i.e.) MLlib, Data Frames, and SQL at the side of Spark Streaming and GraphX. It is allowed to use a mix of those libraries which are coherently associated with the application. The feature of mix streaming, SQL, and complicated analytics, within the same application, makes Spark a general framework.
5. **Advanced Analytics:** Apache Spark also supports "Map" and "Reduce" that has been mentioned earlier. However, at the side of MapReduce, it supports Streaming data, SQL queries, Graph algorithms, and Machine learning. Thus, Apache Spark may be used to perform advanced analytics.

Components of Spark:



1.Spark Core: All the functionalities being provided by Apache Spark are built on the highest of the Spark Core. It delivers speed by providing in-memory

computation capability.

It is the main backbone of the essential I/O functionalities and significant in programming and observing the role of the spark cluster. It holds all the components related to scheduling, distributing and monitoring jobs on a cluster, Task dispatching, Fault recovery. The functionalities of this component are:

1. It contains the basic functionality of spark. (Task scheduling, memory management, fault recovery, interacting with storage systems).
2. Home to API that defines RDDs.

1.Spark SQL Structured data: The Spark SQL component is built above the spark core and used to provide the structured processing on the data.

The main functionality of this module is:

1. It is a Spark package for working with structured data.
2. It Supports many sources of data including hive tablets, parquet, json. 3. It allows the developers to intermix SQK with programmatic data manipulation supported by RDDs in python, scala and java.

2.Spark Streaming: Spark streaming permits ascendible, high-throughput, fault-tolerant stream process of live knowledge streams. Spark can access data from a source like a flume, TCP socket. It will operate different algorithms in which it receives the data in a file system, database and live dashboard. Spark uses Micro-batching for real-time streaming. Micro-batching is a technique that permits a method or a task to treat a stream as a sequence of little batches of information. Hence spark streaming groups the live data into small batches. It delivers it to the batch system for processing. The functionality of this module is:

1. Enables processing of live streams of data like log files generated by production web services.
2. The API's defined in this module are quite similar to spark core RDD API's. **3.Mllib Machine Learning:** MLlib in spark is a scalable Machine learning library that contains various machine learning algorithms. The motive behind MLlib creation is to make the implementation of machine learning simple. It contains machine learning libraries and the implementation of various algorithms. For example, clustering, regression, classification and collaborative filtering. **4.GraphX graph processing:** It is an API for graphs and graph parallel execution.

There is network analytics in which we store the data. Clustering, classification, traversal, searching, and pathfinding is also possible in the graph. It generally optimizes how we can represent vertex and edges in a graph. GraphX also optimizes how we can represent vertex and edges when they are primitive data types. To support graph computation, it supports fundamental operations like subgraph, joins vertices, and aggregate messages as well as an optimized variant of the Pregel API.

Uses of Apache Spark:

1. It is tough to handle the time generated data like log files. Spark is capable enough to work well with streams of information and reuse operations. 2. As spark is capable of storing information in memory and might run continual queries quickly.

RDD vs. DataFrame vs. Dataset Differences

difference between RDD vs DataFrame vs DataSet API in Spark:

	RDD	DataFrame	Dataset
Release version	Spark 1.0	Spark 1.3	Spark 1.6
Data Representation	Distributed collection of elements.	Distributed collection of data organized into columns.	Combination of RDD and DataFrame.
Data Formats	Structured and unstructured are accepted.	Structured and semi structured are accepted.	Structured and unstructured are accepted.
Data Sources	Various data sources.	Various data sources.	Various data sources.
Immutability and Interoperability	Immutable partitions that easily transform into DataFrames.	Transforming into a Dataframe loses the original RDD.	The original RDD regenerates after transformation.
Compile-time type safety	Available compile-time type safety.	No compile-time type safety. Errors detect on runtime.	Available compile time type safety.

Optimization	No built-in optimization engine. Each RDD is optimized individually.	Query optimization through the Catalyst optimizer.	Query optimization through the Catalyst optimizer, like DataFrames.
Serialization	RDD uses Java serialization to encode data and is expensive. Serialization requires sending both the data and structure between nodes.	There is no need for Java serialization and encoding. Serialization happens in memory in binary format.	Encoder handles conversions between JVM objects and tables, which is faster than Java serialization.
Garbage Collection	Creating and destroying individual objects creates garbage collection overhead.	Avoids garbage collection when creating or destroying objects.	No need for garbage collection
Efficiency	Efficiency decreased for serialization of individual objects.	In-memory serialization reduces overhead. Operations performed on serialized data without the need for deserialization.	Access to individual attributes without deserializing the whole object.
Lazy Evaluation	Yes.	Yes.	Yes.
Programming Language Support	Java Scala Python R	Java Scala Python R	Java Scala
Schema Projection	Schemas need to be defined manually.	Auto-discovery of file schemas.	Auto-discovery of file schemas.
Aggregation	Hard, slow to perform simple aggregations and grouping operations.	Fast for exploratory analysis. Aggregated statistics on large datasets are possible and perform quickly.	Fast aggregation on numerous datasets.

What is an RDD?