Databricks Certified Associate Developer for Apache Spark

By Aviral Bhardwaj

WhatsApp- +91-9307854232

RDD and DataFrame and Dataset

What are they?

1. **They all are APIs provided by Spark for developers for data processing and analytics**
2. **Users can choose any of the API while working with Spark**
3. **In terms of functionality, all are same and returns same output for provided input data. But they differ in the way of handling and processing data. So, there is difference in terms of performance, user convenience and language support etc.**

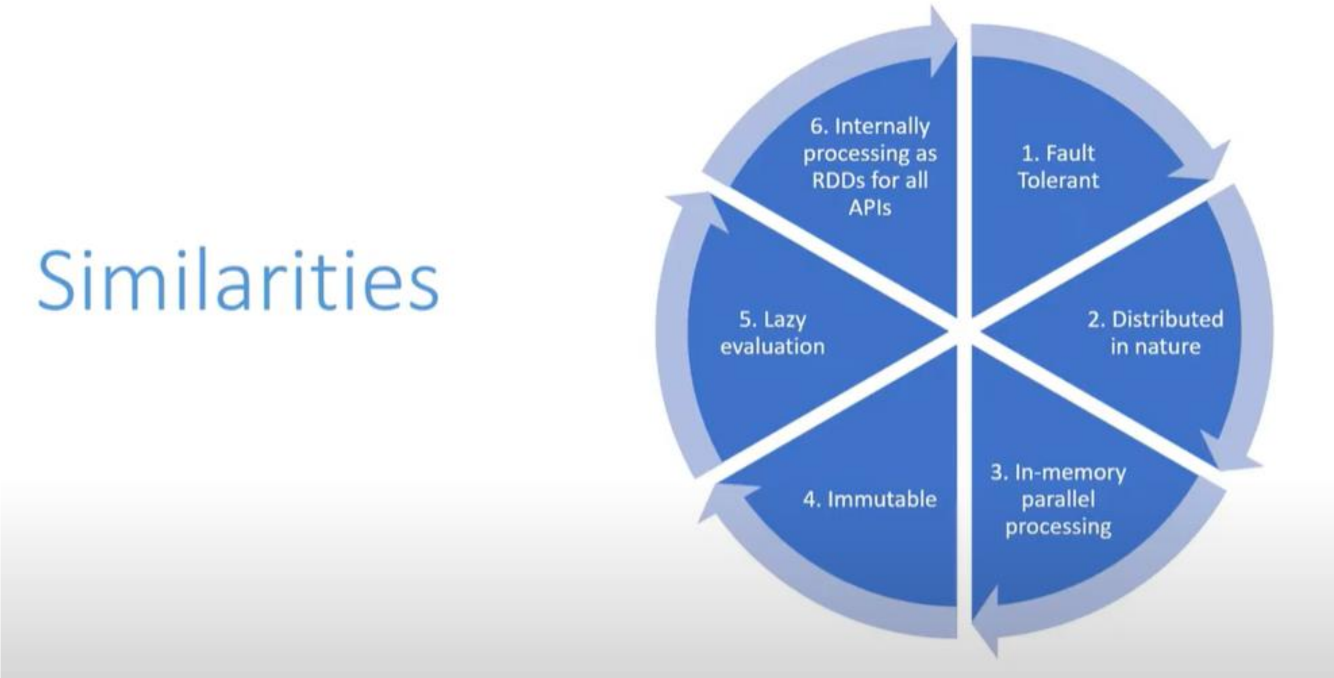
Journey Of API

RDD is Introduced in 2011 (Spark 1.0)

Dataframe is Introduced in 2013 (Spark 1.3)

DataSet is Introduced in 2015 (Spark 1.6)

DataSet= Best of RDD (Programming Control OOPS and Type Safety)+Best Of DataFrame(Relational Format and Optimization and Memory Management)



Similarities

1. Fault Tolerant

2. Distributed in nature

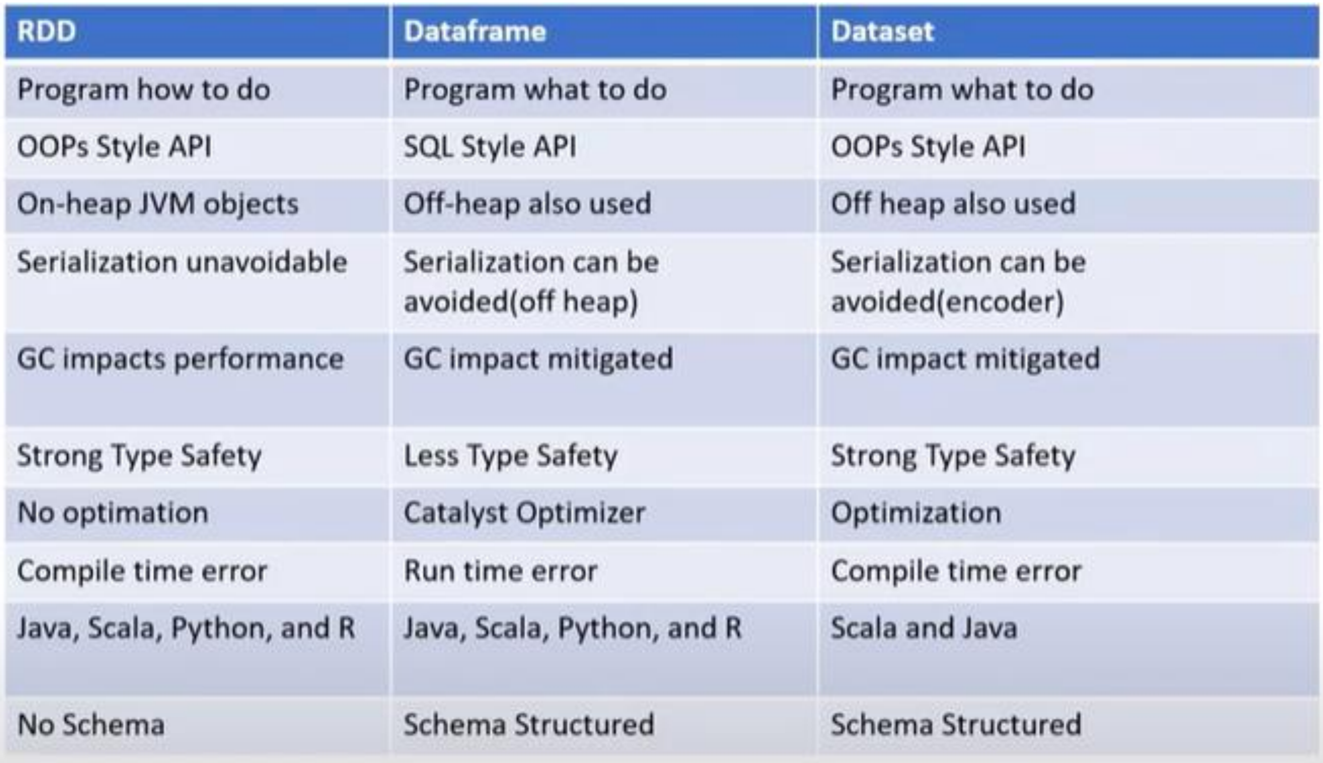
3. In-memory parallel processing

4. Immutable

5. Lazy evaluation

6. Internally processing as RDDs for all APIs

Similarities



RDD	Dataframe	Dataset
Program how to do	Program what to do	Program what to do
OOPs Style API	SQL Style API	OOPs Style API
On-heap JVM objects	Off-heap also used	Off heap also used
Serialization unavoidable	Serialization can be avoided(off heap)	Serialization can be avoided(encoder)
GC impacts performance	GC impact mitigated	GC impact mitigated
Strong Type Safety	Less Type Safety	Strong Type Safety
No optimation	Catalyst Optimizer	Optimization
Compile time error	Run time error	Compile time error
Java, Scala, Python, and R	Java, Scala, Python, and R	Scala and Java
No Schema	Schema Structured	Schema Structured

Some Important Concepts again

Job→Stage→Tasks

* **Driver and worker Process:** These are nothing but JVM process. Within one worker node, there could be multiple executors. Each executor runs its own JVM process.

* **Application:** It could be single command or combination of multiple notebooks with complex logic. When code is submitted to spark for execution, Application starts.

Job- When an application is submitted to Spark, driver process converts the code into job.

* **Stage:** Jobs are divided into stages. If the application code demands shuffling the data across nodes, new stage is created. Number of stages are determined by number of shuffling operations. Join is example of shuffling operation

* **Tasks:** Stages are further divided into multiple tasks. In a stage, all the tasks would execute same logic. Each task will process 1 partition at a time. So number of partition in the distributed cluster determines the number of tasks in each stage

* **Transformation:** transforms the input RDD and creates new RDD. Until action is called, transformations are evaluated lazily

* **DAG:** Directed Acyclic Graph keeps track of all transformation. For each transformation, logical plan is created and lineage graph is maintained by DAG

* **Action:** When data output is needed for developer or for storage purpose, action is called. Action would be executed based on DAG and processes the actual data

* **RDD:** Resilient Distributed Dataset is basic data structure of Spark. When spark reads or creates data, it creates RDD which is distributed across nodes in the form of partition.

* **Executor:** Each worker node can consists of many executors. It can be configured by spark settings

* **Partition:** RDD/Dataframe is stored in-memory of cluster in the form of partition

* **Core:** Each executor can consists of multiple cores. This is configurable by spark settings. Each core(if single thread) can process one task at a time.

* **On-heap memory:** The executor memory that lies within JVM process managed JVM

* **Off-heap memory:** The executor memory that lies outside JVM process managed by OS