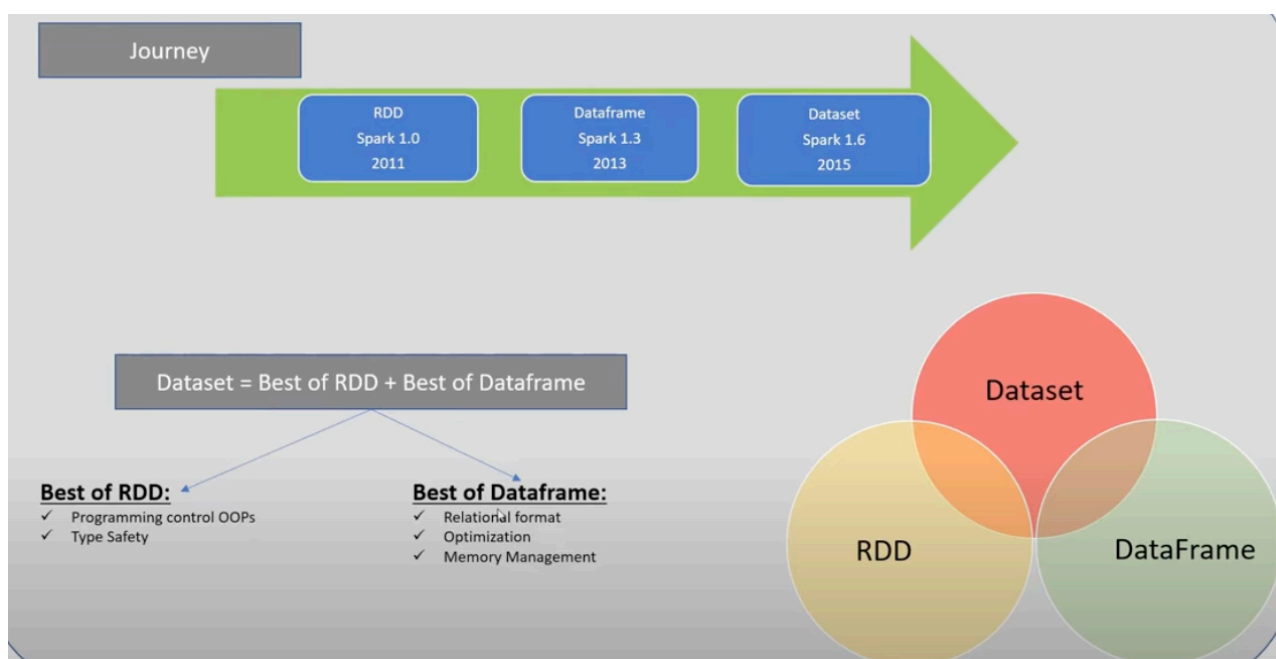
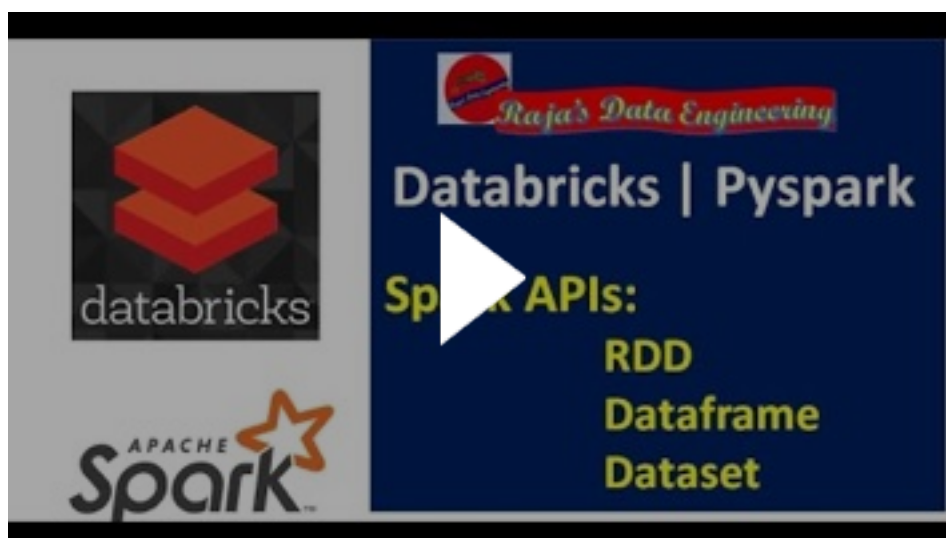


02. Databricks | PySpark: RDD, Dataframe and Dataset

Monday, 4 November 2024

10:30 AM

[02. Databricks | PySpark: RDD, Dataframe and Dataset](#)



In RDD -

Programing has OOPS style programing

Type safety - means all columns have certain data types, this helps in avoiding runtime errors.

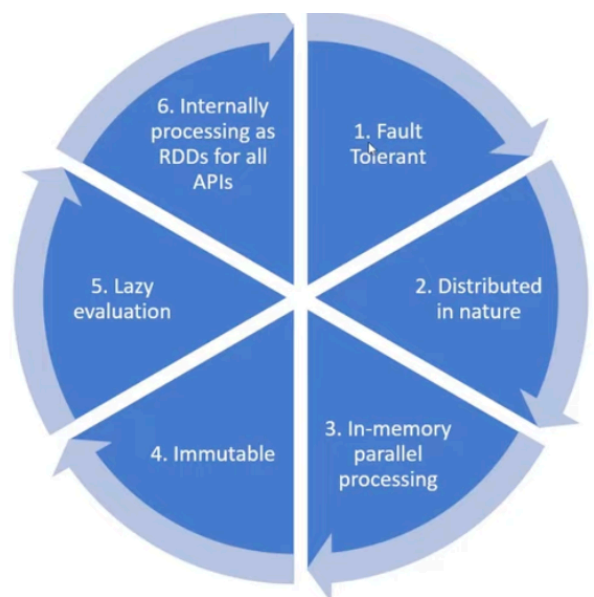
In Dataframe -

Structured relational format of storing data

Optimization, since stored data has a schema, optimizer can collect statistics in dataframe and based on those statistics it can create different N number of logical plans and then choose the best for physical plan. That is how dataframe can be optimized as opposed to RDD.

In RDD all objects were stored on heap memory in the form of JVM objects, in dataframe objects can be stored off-heap memory that helps in memory management.

Similarities



When we run transformation in spark(using any API), it do not run it immediately, it creates a logical plan and based on that it will construct a lineage graph called DAG, if there is some failure in middle of process these APIs can construct the data again using the graph as graph has all the flow stored in it.


Data is stored in the form of multiple partitions across multiple nodes making it possible to do parallel processing.

Spark has in-memory computation , means data will be brought to RAM and then transformed upon instead of traditional way of keeping it in hard disk.

Immutable - source data cannot be modified once stored in spark, when transformation is run, new copy of data is created and old one is also kept.

Lazy evaluation - data processing do not happen immediately, for logical plan is created then DAG and then processing happens.

Dataframe of dataset is converted internally by spark internally into RDD.



| RDD | Dataframe | Dataset |
|--------------------------------|--|---------------------------------------|
| Program how to do ₂ | 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 |

When programming using RDD, we need to tell what to do and how to do. But in dataframe and dataset we do not have to tell how to do as it has optimizer, it will automatically handle the how to do part by creating the best logical plan.

GC is garbage collector - in RDD when memory is full GC has to run to remove obsolete JVM objects that are taking up memory but not used.

This do not happen in dataframe of Dataset as off-heap memory is used.

In dataframe less type safety- in case of datatype miss match it will throw error at run time, but in dataset this happens at compilation only hence its faster.

Dataset supports only Scala and Java

Dataframe support - Java, scala , Python, and R