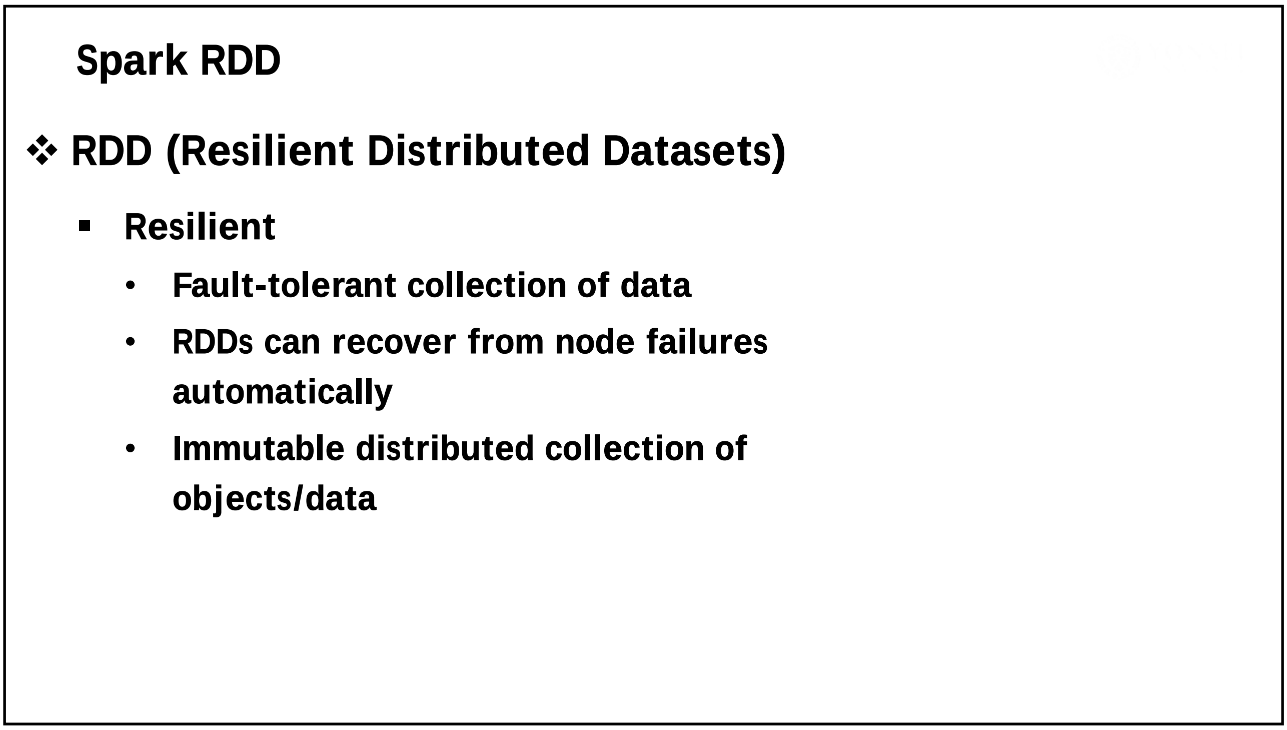
Big Data

Spark RDD

Spark RDD

❖ RDD (Resilient Distributed Datasets)

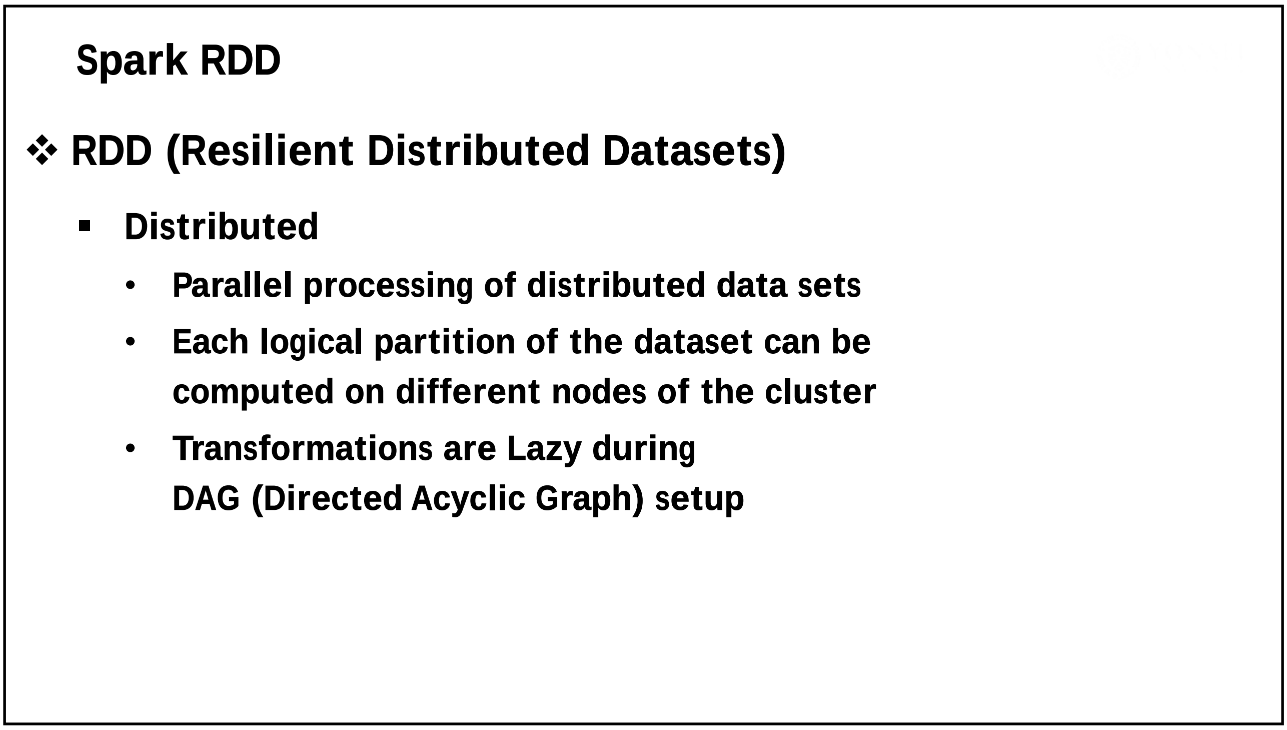
- RDD General Characteristics
  - Sequence of deterministic Transformations & Actions are conducted on a
    - Stored dataset
    - Another RDD
    - Group of RDDs
  - RDD = Resilient + Distributed + Datasets



Spark RDD

❖ **RDD (Resilient Distributed Datasets)**

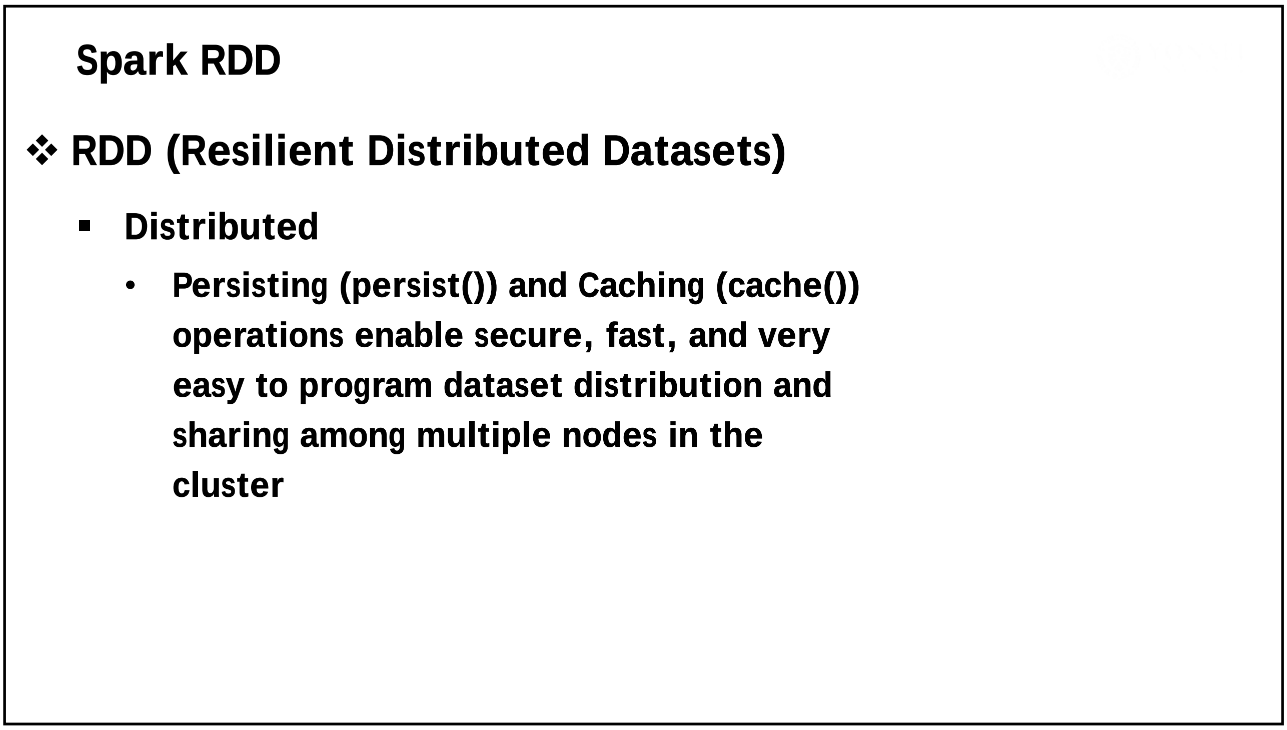
- **Resilient**
  - **Fault-tolerant collection of data**
  - **RDDs can recover from node failures automatically**
  - **Immutable distributed collection of objects/data**



Spark RDD

❖ **RDD (Resilient Distributed Datasets)**

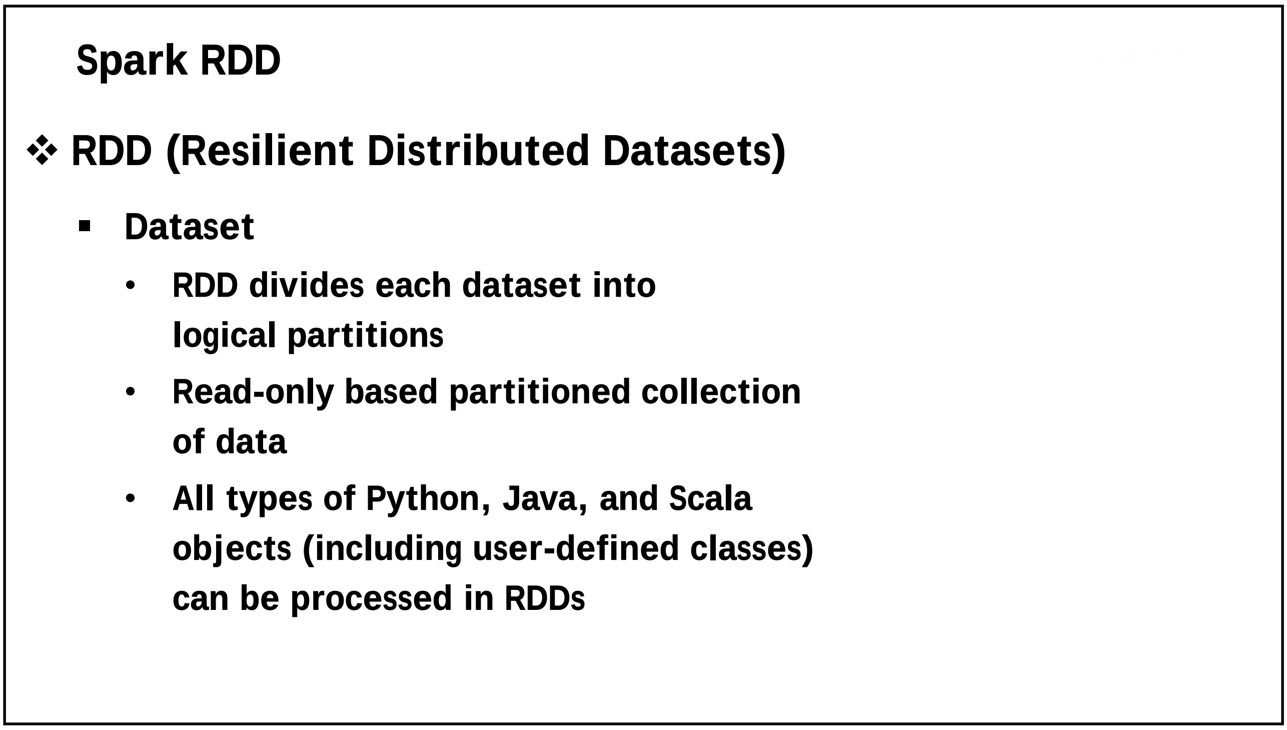
- **Distributed**
  - **Parallel processing of distributed data sets**
  - **Each logical partition of the dataset can be computed on different nodes of the cluster**
  - **Transformations are Lazy during DAG (Directed Acyclic Graph) setup**



Spark RDD

❖ RDD (Resilient Distributed Datasets)

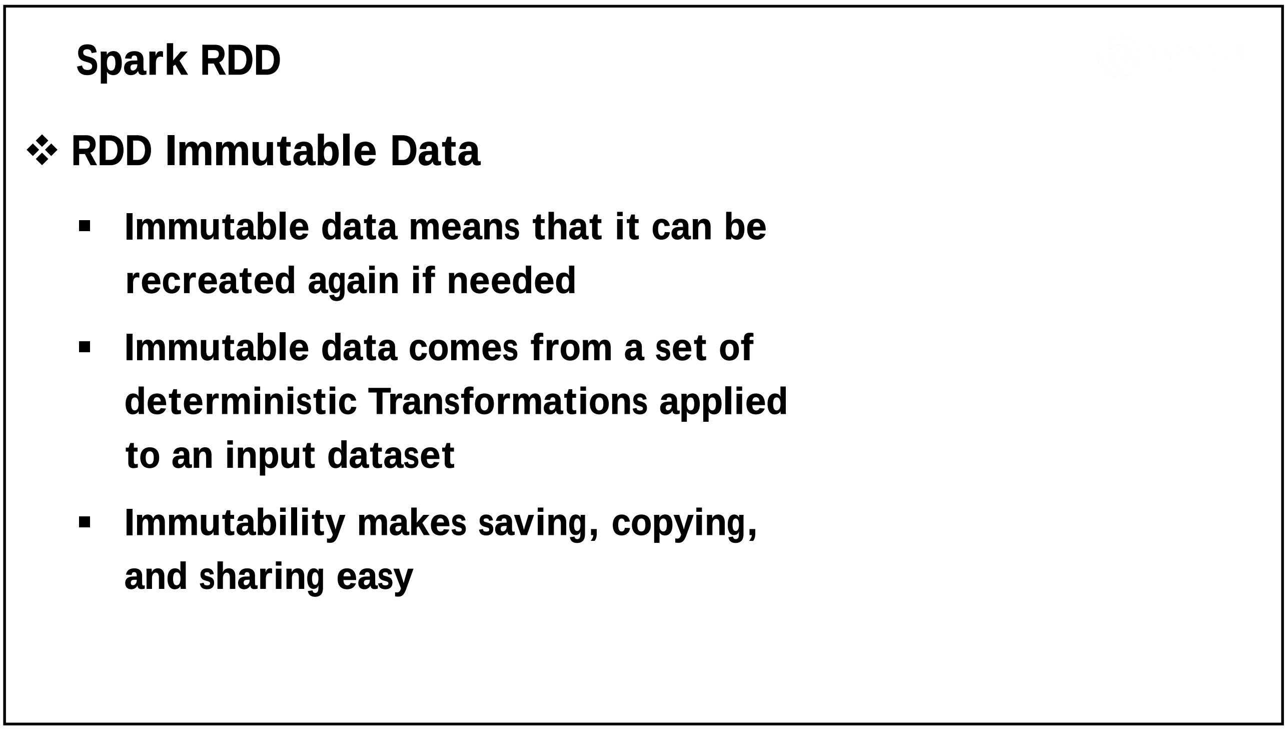
- Distributed
  - Persisting (persist()) and Caching (cache()) operations enable secure, fast, and very easy to program dataset distribution and sharing among multiple nodes in the cluster



Spark RDD

❖ RDD (Resilient Distributed Datasets)

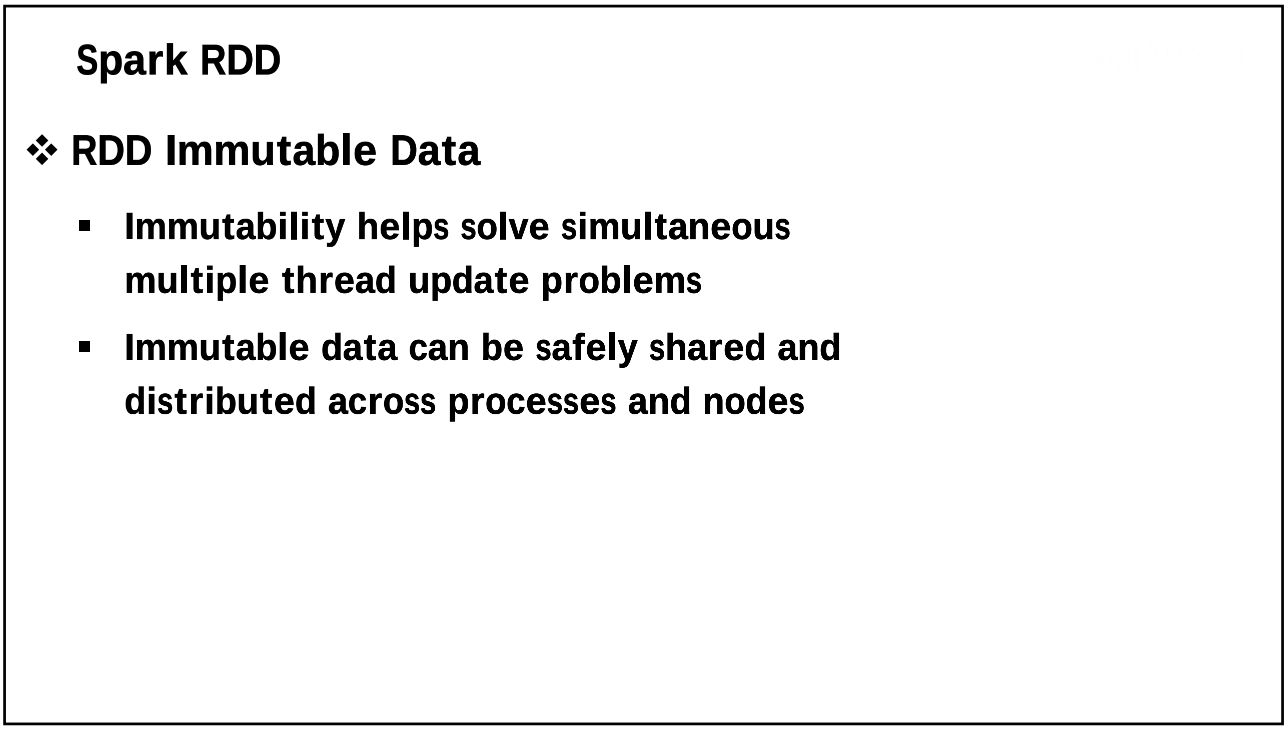
- Dataset
  - RDD divides each dataset into logical partitions
  - Read-only based partitioned collection of data
  - All types of Python, Java, and Scala objects (including user-defined classes) can be processed in RDDs



Spark RDD

❖ RDD Immutable Data

- Immutable data means that it can be recreated again if needed
- Immutable data comes from a set of deterministic Transformations applied to an input dataset
- Immutability makes saving, copying, and sharing easy



Spark RDD

❖ RDD Immutable Data

- Immutability helps solve simultaneous multiple thread update problems
- Immutable data can be safely shared and distributed across processes and nodes

Spark RDD

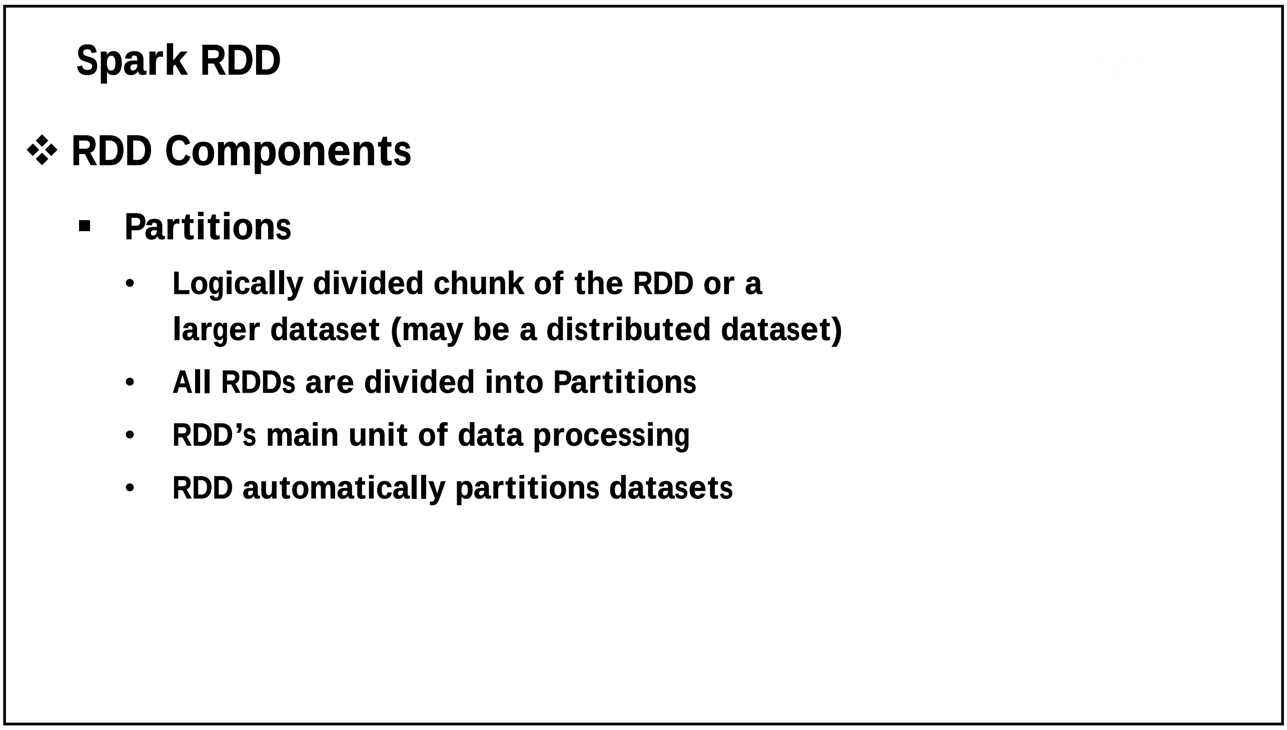
❖ RDD Persistence

- **Persisting (or Caching) a dataset in memory enhances data processing speed across multiple operations**
- **Persisted RDD makes a node(s) store its Transformed data in its partition memory to quickly reuse it for other Actions processed on that dataset**

Spark RDD

❖ RDD Persistence

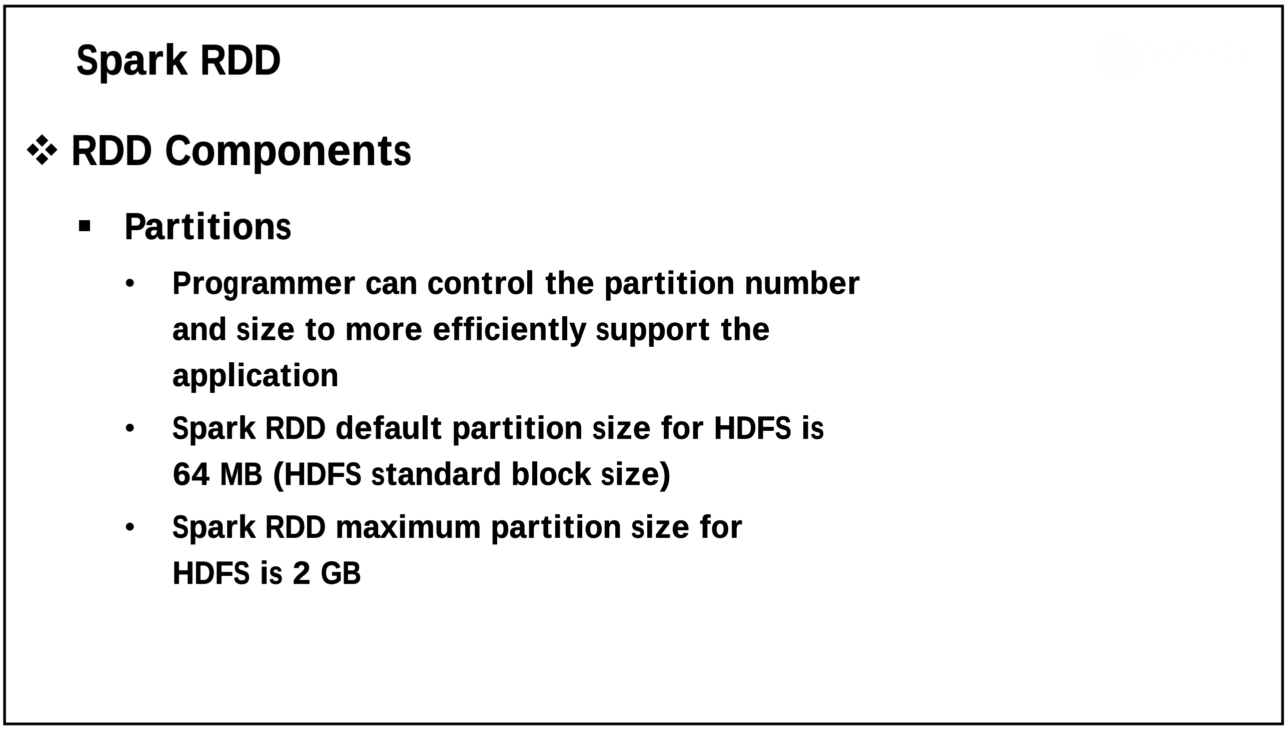
- **Using a DAG (Directed Acyclic Graph) and having Lazy Transformations helps to schedule optimized RDD Persisting for multiple Actions to be executed more efficiently**



Spark RDD

❖ RDD Components

- **Partitions**
  - **Logically divided chunk of the RDD or a larger dataset (may be a distributed dataset)**
  - **All RDDs are divided into Partitions**
  - **RDD’s main unit of data processing**
  - **RDD automatically partitions datasets**



Spark RDD

❖ RDD Components

- **Partitions**
  - **Programmer can control the partition number and size to more efficiently support the application**
  - **Spark RDD default partition size for HDFS is 64 MB (HDFS standard block size)**
  - **Spark RDD maximum partition size for HDFS is 2 GB**

Spark RDD

❖ RDD Components

- **Metadata & Data Type**
  - **Data and object type considered in the RDD partitioning and data placement**
  - **Structured data, Semi-structured data, and Unstructured data**
- **Task & Transformation**
  - **Function applied to process the data partition**
  - **Parent RDD ➔ Function ➔ Child RDD**
    - **Parent : Child ➔ 1:1, N:1, 1:M, N:M**

Spark RDD

❖ RDD Types

- **HadoopRDD**
  Each HDFS block is a partition
- **FilteredRDD**
  Child RDD partitions are same as the parent RDD partitions
- **JoinedRDD**
  One partition per Reduce task
- **MappedRDD**
- **PairRDD**
- **ShuffledRDD**
- **UnionRDD**
- **PythonRDD**
- **DoubleRDD**
- **JdbcRDD**
- **JsonRDD**
- **SchemaRDD**
- **VertexRDD**
- **EdgeRDD**
- **CassandraRDD**
- **GeoRDD**
- **EsSpark**
  ElasticSearch
- **etc.**

Spark RDD

❖ SchemaRDD

- RDD that has an associated schema
- Has all standard RDD functions and also can be used in relational queries

Spark RDD

❖ SchemaRDD creation methods

- Loading data in from external sources to SchemaRDD form
- Methods of converting a standard RDD into a SchemaRDD
- Importing a SQLContext
- Apply the createSchemaRDD function on SQLContext

Spark RDD

❖ SchemaRDD

- SchemaRDD can be registered as a table in the SQLContext it was imported from
- SchemaRDD registered as a table can be used in SQL statements

Big Data

References

References

- Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia, *Learning Spark: Lightning-Fast Big Data Analysis*. 1st Edition. O'Reilly, 2015.
- Sameer Farooqui, Databricks, **Advanced Apache Spark Training**, Devops Advanced Class, Spark Summit East 2015, http://slideshare.net/databricks, www.linkedin.com/in/blueplastic, March 2015.
- Apache Spark documents (all documents and tutorials were used)
  - http://spark.apache.org/docs/latest/rdd-programming-guide.html
  - http://spark.apache.org/docs/latest/rdd-programming-guide.html#working-with-key-value-pairs
  - https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html#rdd-persistence
- Wikipedia, www.wikipedia.org
- Stackoverflow, https://stackoverflow.com/questions
- Bernard Marr, "Spark Or Hadoop -- Which Is The Best Big Data Framework?," Forbes, Tech, June 22, 2015.
- Quick introduction to Apache Spark, https://www.youtube.com/watch?v=TgiBvKcGL24
- Wide vs Narrow Dependencies, https://github.com/rohgar/scala-spark-4/wiki/Wide-vs-Narrow-Dependencies

References

- Partitions and Partitioning, https://jaceklaskowski.gitbooks.io/mastering-apache-spark/spark-rdd-partitions.html
- Neo4j, "From Relational to Neo4j," https://neo4j.com/developer/graph-db-vs-rdbms/ (last accessed Jan. 1, 2018).

Image Sources

- By Robivy64 at English Wikipedia [Public domain], via Wikimedia Commons
- Teravolt at English Wikipedia [CC BY 3.0 (http://creativecommons.org/licenses/by/3.0)], via Wikimedia Commons
- By Konradr (Own work) [GFDL (http://www.gnu.org/copyleft/fdl.html) or CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/)], via Wikimedia Commons