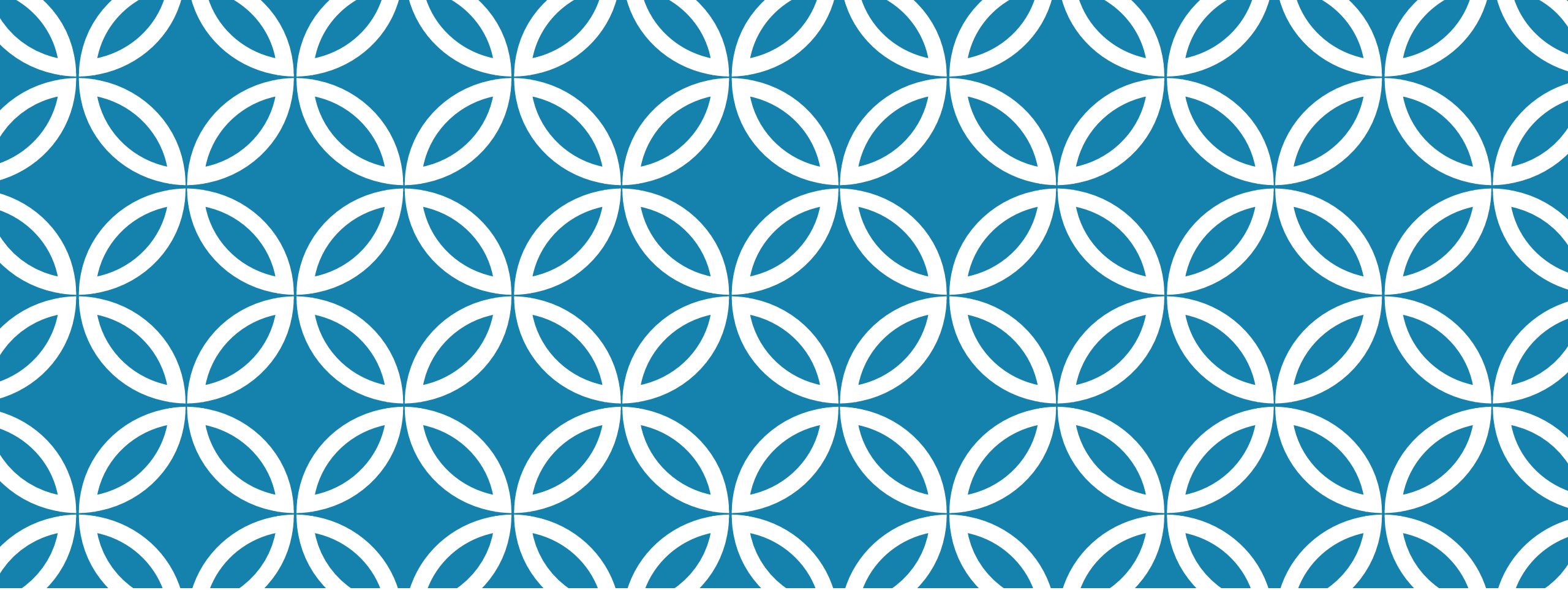


SPARK BIG DATA

Dr. Salahuddin Shaikh

SPARK BIG DATA

- Spark has been proposed by Apache Software Foundation to speed up the software process of Hadoop computational computing. Spark includes its cluster management, while Hadoop is only one of the forms for implementing Spark.
- Basically Spark is a framework – in the same way that Hadoop is – which provides a number of inter-connected platforms, systems and standards for Big Data projects.
- Like Hadoop, Spark is open-source and under the wing of the Apache Software Foundation. Essentially, open-source means the code can be freely used by anyone.

SPARK BIG DATA

- Spark applies Hadoop in two forms. The first form is **storage** and another one is **processing**. Thus, Spark includes its computation for cluster management and applies Hadoop for only storage purposes.
- Spark is seen by techies in the industry as a more advanced product than Hadoop – it is newer, and designed to work by processing data in chunks "in memory". This means it transfers data from the physical, magnetic hard discs into far-faster electronic memory where processing can be carried out far more quickly – up to 100 times faster in some operations.

APACHE SPARK

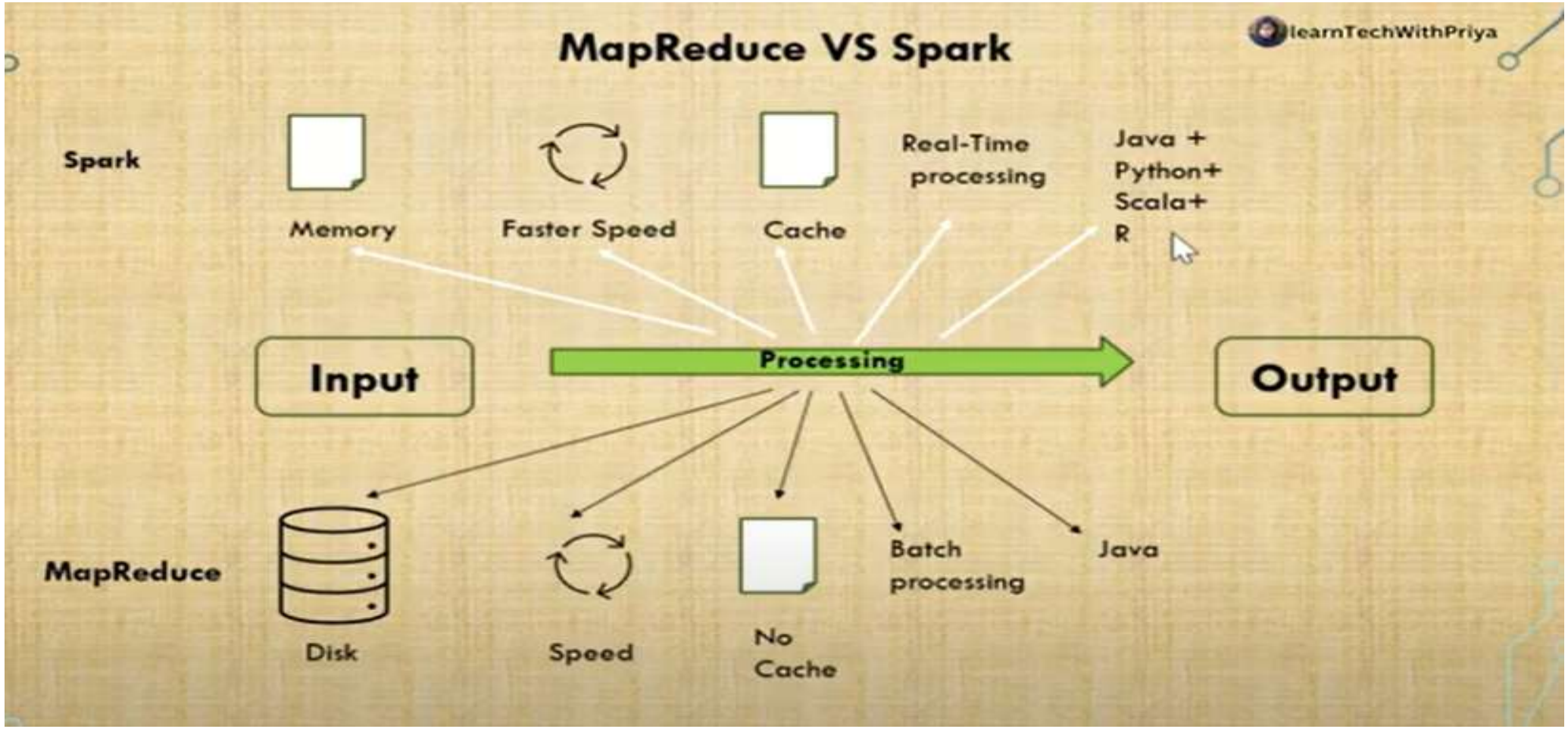
Apache Spark is a **distributed** and **open-source processing system.** It is used for the workloads of 'Big data'. Spark utilizes optimized query execution and in-memory caching for rapid queries across any size of data. It is simply a general and fast engine for much large-scale processing of data.

It is much faster as compared to the previous concepts to implement with Big Data such as classical **MapReduce.** Spark is faster die to it executes on RAM/memory and enables the processing faster as compared to the disk drivers.

APACHE SPARK

- Spark is simple due to it could be used for more than one thing such as working with data streams or graphs, Machine Learning algorithms, inhaling data into the database, building data pipelines, executing distributed SQL, and others.
- Another element of the framework is Spark Streaming, which allows applications to be developed which perform analytics on streaming, real-time data – such as automatically analyzing video or social media data – on-the-fly, in real-time.

APACHE SPARK & MAP REDUCE



APACHE SPARK EVOLUTION

- Spark is one of the **most important sub-projects of Hadoop**. It was developed in APMLab of UC Berkeley in 2009 by Matei Zaharia. In 2010, it was an open-source under the BSD license. Spark was donated in 2013 to the Apache Software Foundation. Apache Spark is now a top-level project of Apache from 2014 February.
- Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools.

FEATURES OF APACHE SPARK

Apache Spark has following features.

Speed – Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.

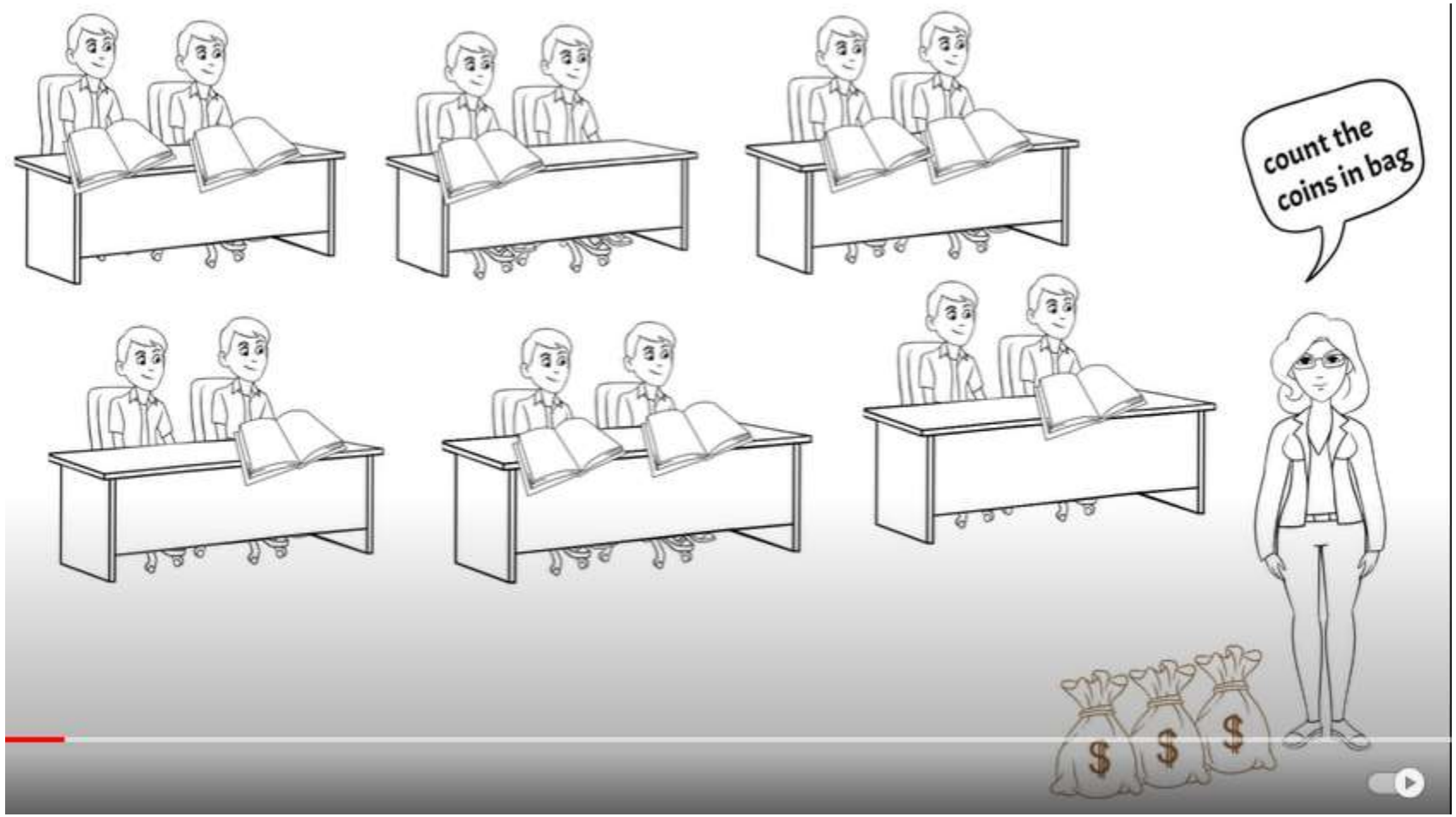
Supports multiple languages – Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.

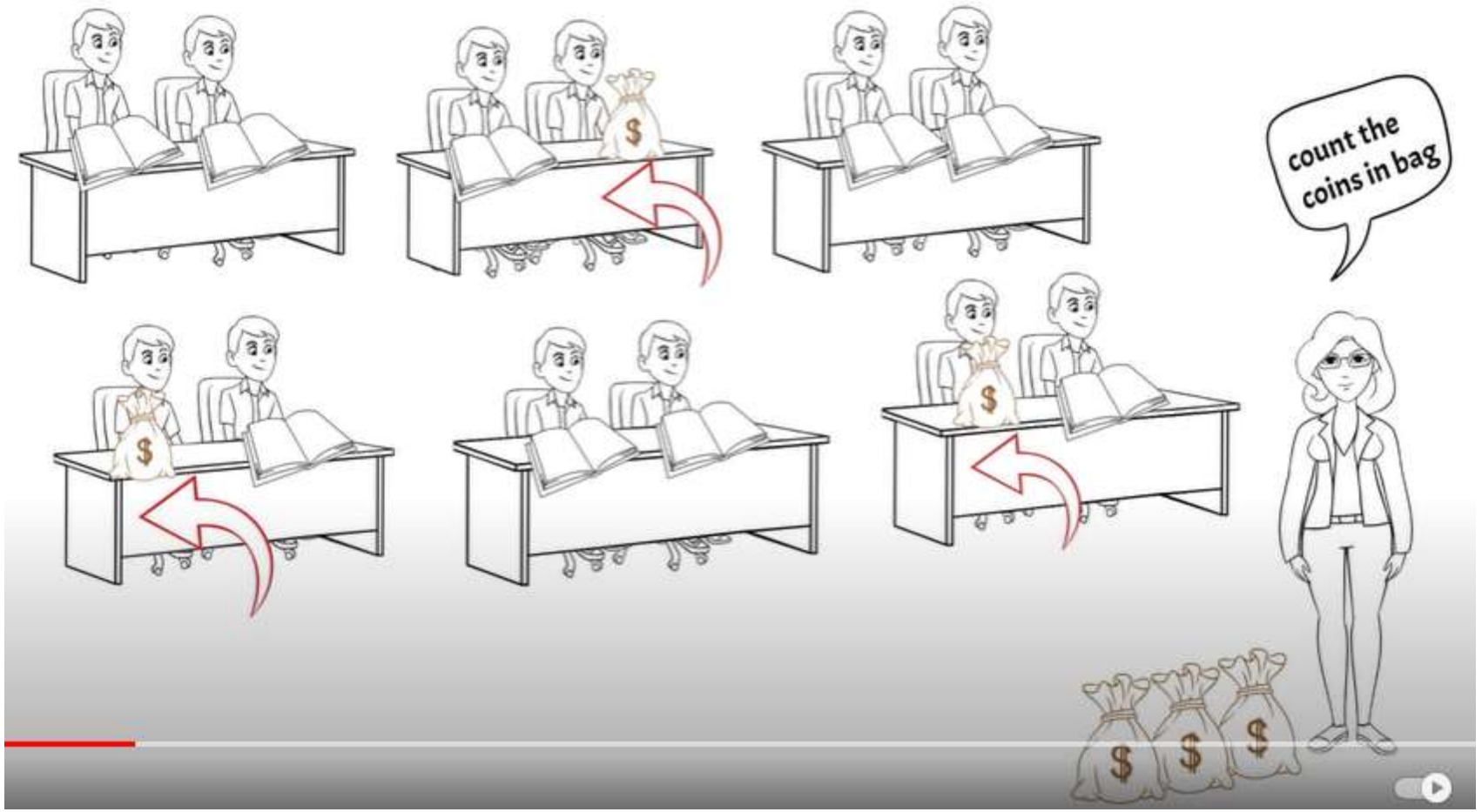
Advanced Analytics – Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

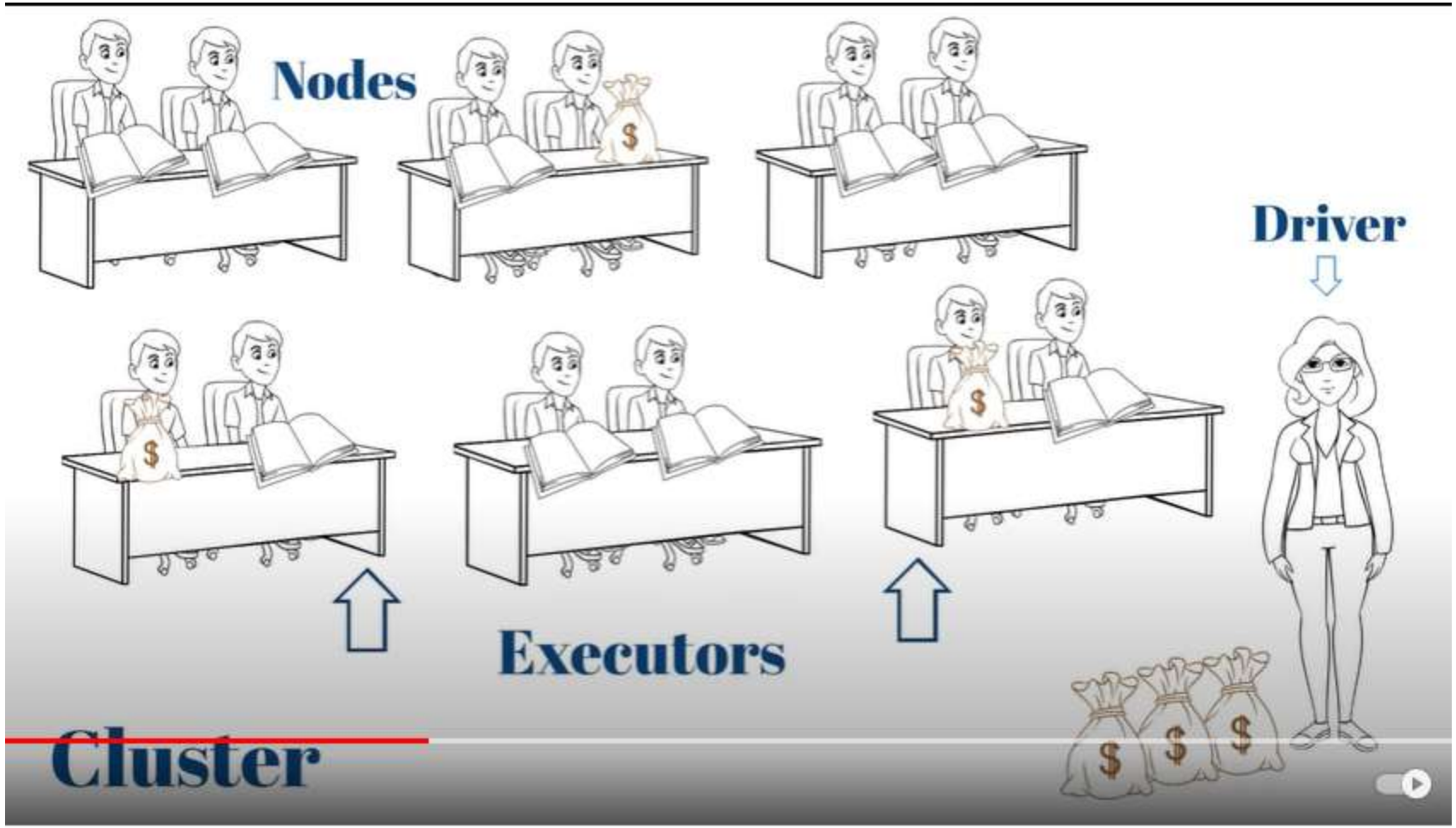
Components

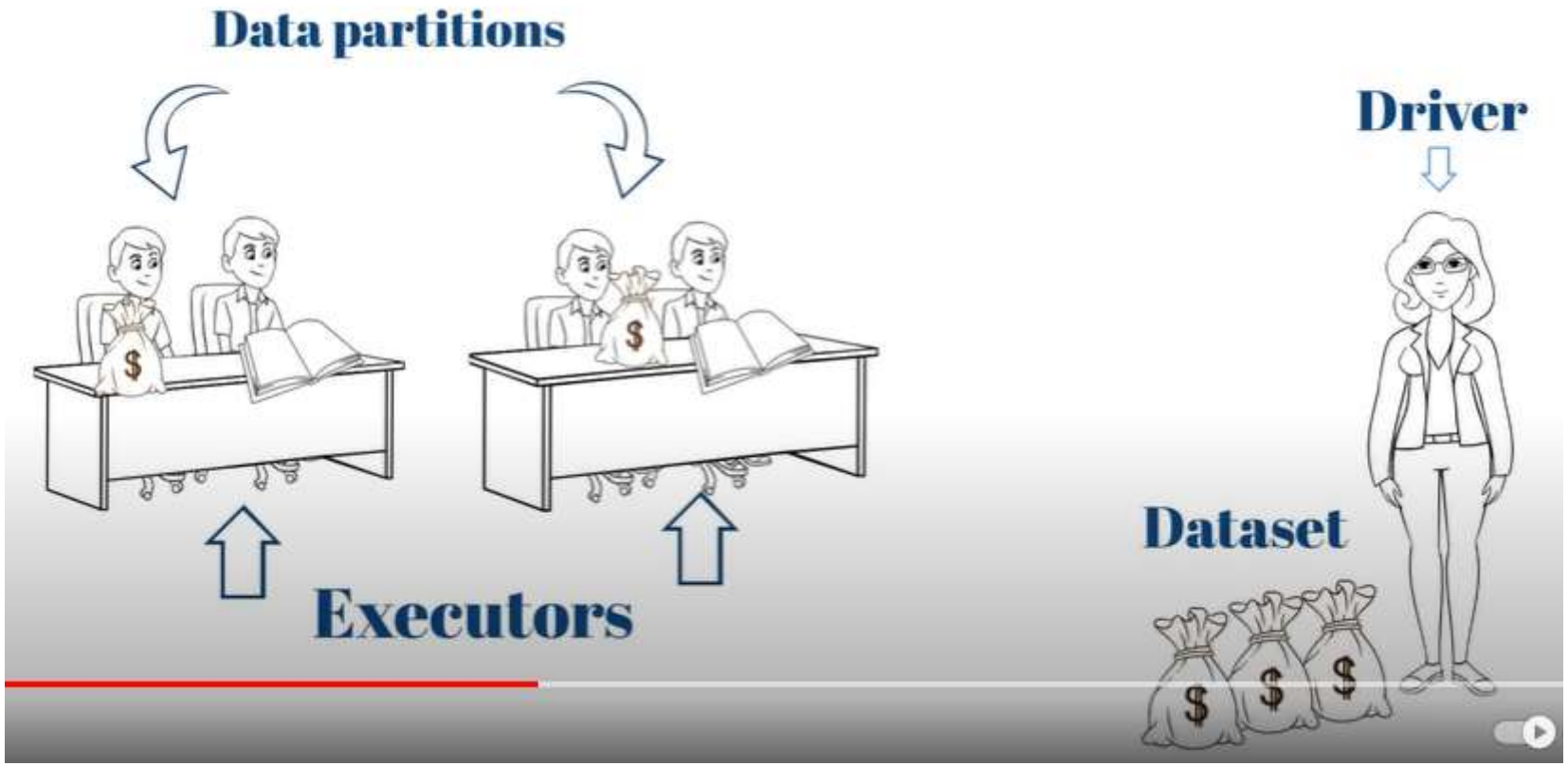
Driver
Executor
Nodes
Cluster

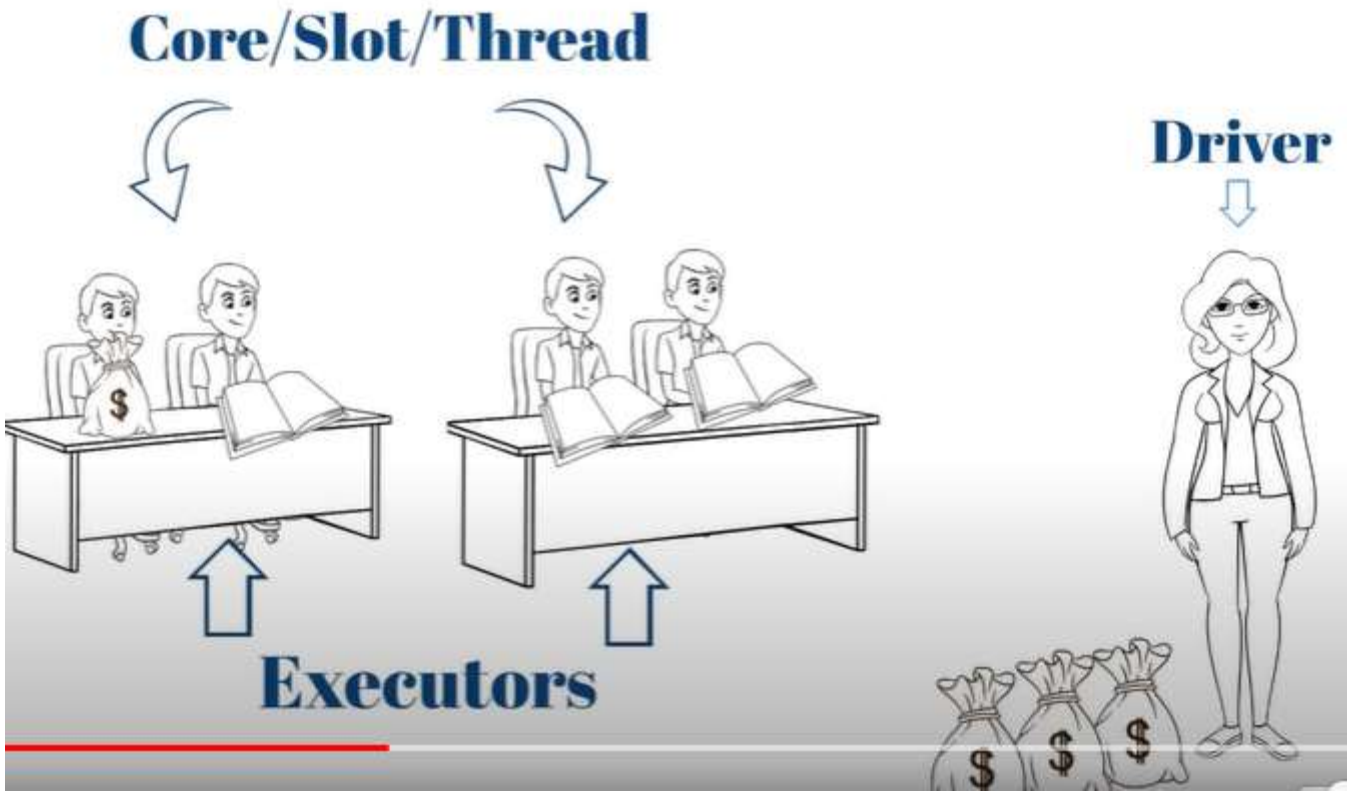
Core/Slot/Thread
Data set
Data Partitions

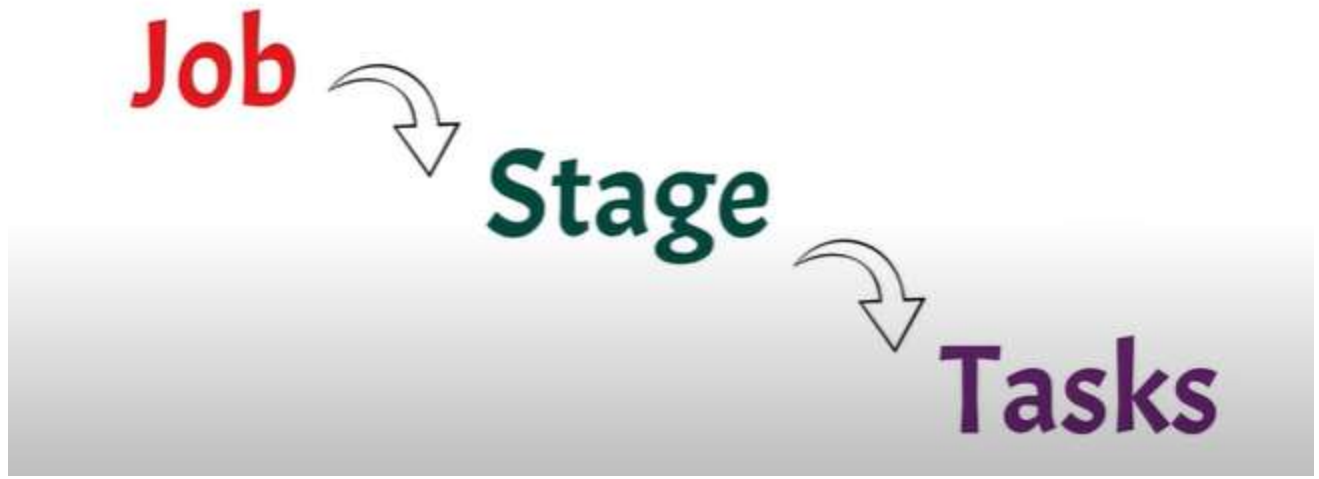

count the
coins in bag
$
$
$

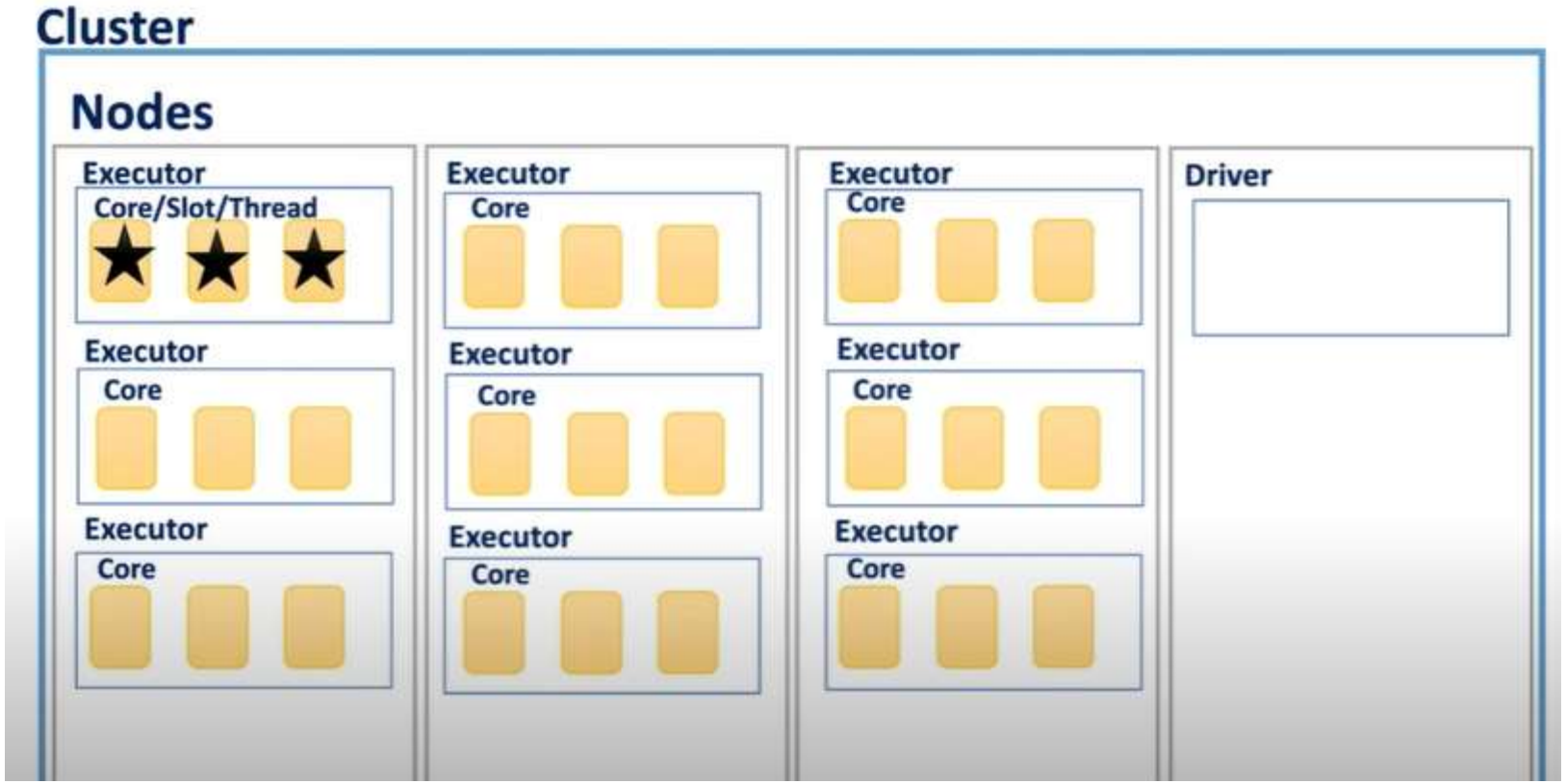

count the
coins in bag


Nodes
Driver
Executors
Cluster


Data partitions
Driver
Dataset
Executors

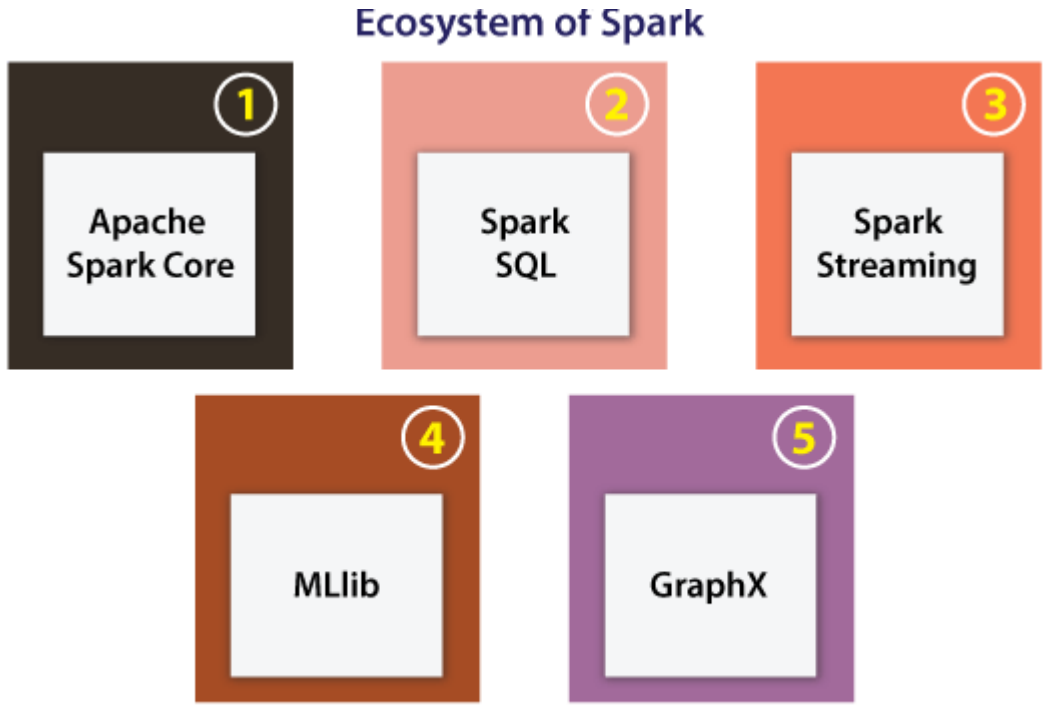

Core/Slot/Thread
Driver
Executors


Job
Stage
Tasks


Cluster
Nodes
Executor
Core/Slot/Thread
Executor
Core
Executor
Core
Executor
Core
Executor
Core
Executor
Core
Executor
Core
Executor
Core
Executor
Core
Executor
Core
Driver

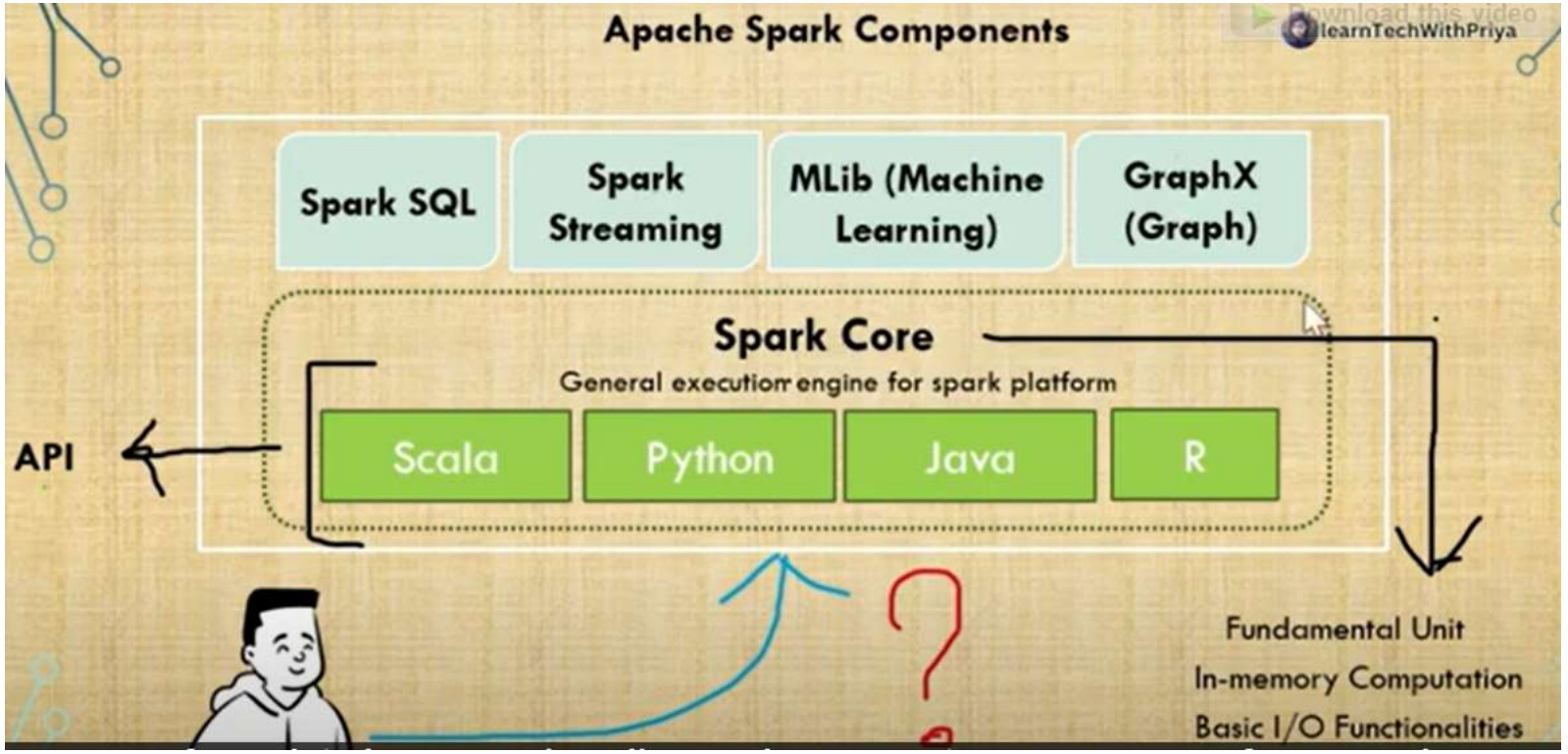
ECOSYSTEM OF SPARK

The main components of the Spark Ecosystem are explained below:



ECOSYSTEM OF SPARK

The main components of the Spark Ecosystem are explained below:



ECOSYSTEM OF SPARK

Apache Spark Core: Apache Spark Core can be defined as an underlying normal execution engine for the platform of Spark. It facilitates referencing data sets and in-memory computing within the external storage structures.

Spark SQL: This component is a module of Apache Spark for operating with many kinds of structured data. Various interfaces provided by Spark SQL facilitates Spark along with a lot of information regarding both the computation and data being implemented.

Spark Streaming: Spark streaming permits Spark for processing streaming data in **real-time**. The data could be inhaling from several sources such as Hadoop Distributed File System (HDFS), Flume, and Kafta. After that data could be processed with complex algorithms and then pushed out towards the live dashboards, databases, and file systems.

ECOSYSTEM OF SPARK

Machine Learning Library (MLlib): Apache Spark is armed with a prosperous library called MLlib. The MLlib includes a wide range of machine learning algorithms collaborative filtering, clustering, regression, and classifications. Also, it contains other resources for tuning, evaluating, and constructing ML pipelines. Each of these functionalities supports Spark scale-out around the cluster.

GraphX: Apache Spark comes using a library for manipulating graph databases and implement computations known as GraphX. This component unifies Extract, Transform, and Load (ETL) process, constant graph computation, and exploratory analysis in an individual system.

ARCHITECTURE OF SPARK

The architecture of Spark contains three of the main elements which are listed below:

API

Data Storage

Resource Management

Let's defines these elements in detail.

API

This element facilitates many developers of the applications for creating Spark-based applications with a classic API interface. Spark offers API for Python, Java, and Scala programming languages.

ARCHITECTURE OF SPARK

Data Storage

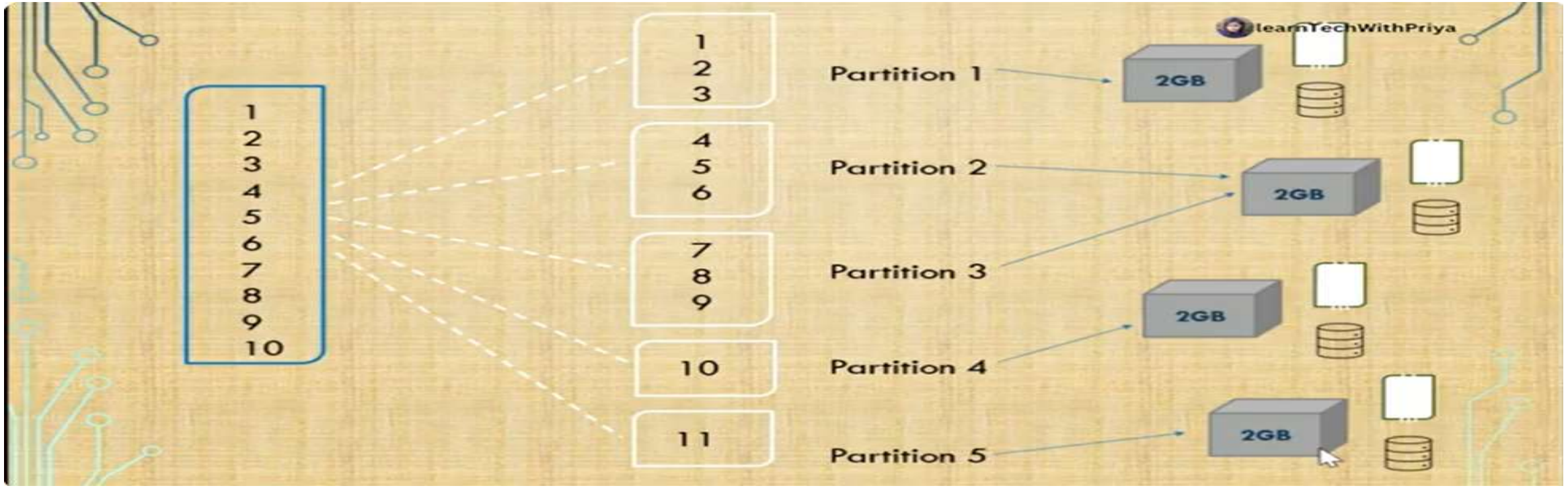
Spark applies the Hadoop Distributed File System for various purposes of data storage. It works with any data source that is compatible with Hadoop including Cassandra, HBase, HDFS, etc.

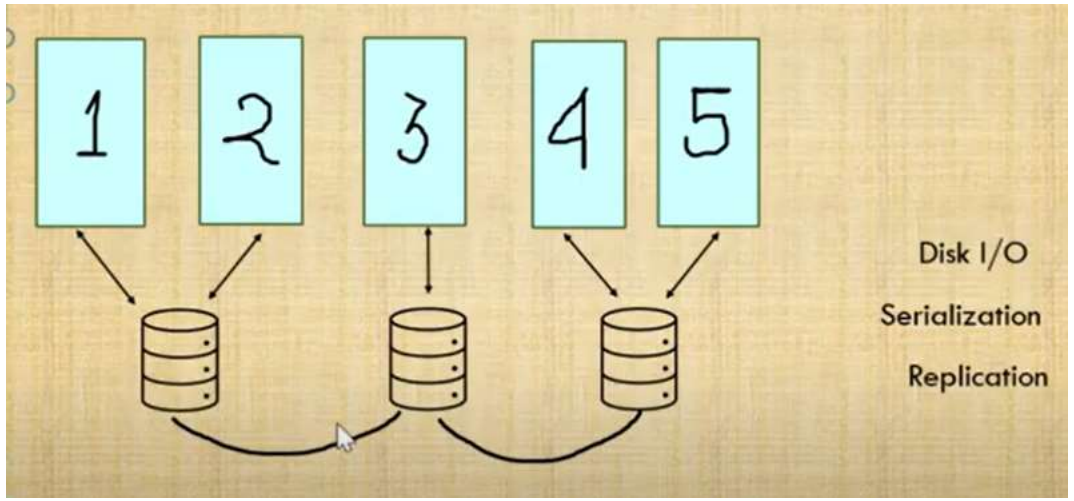
Resource Management

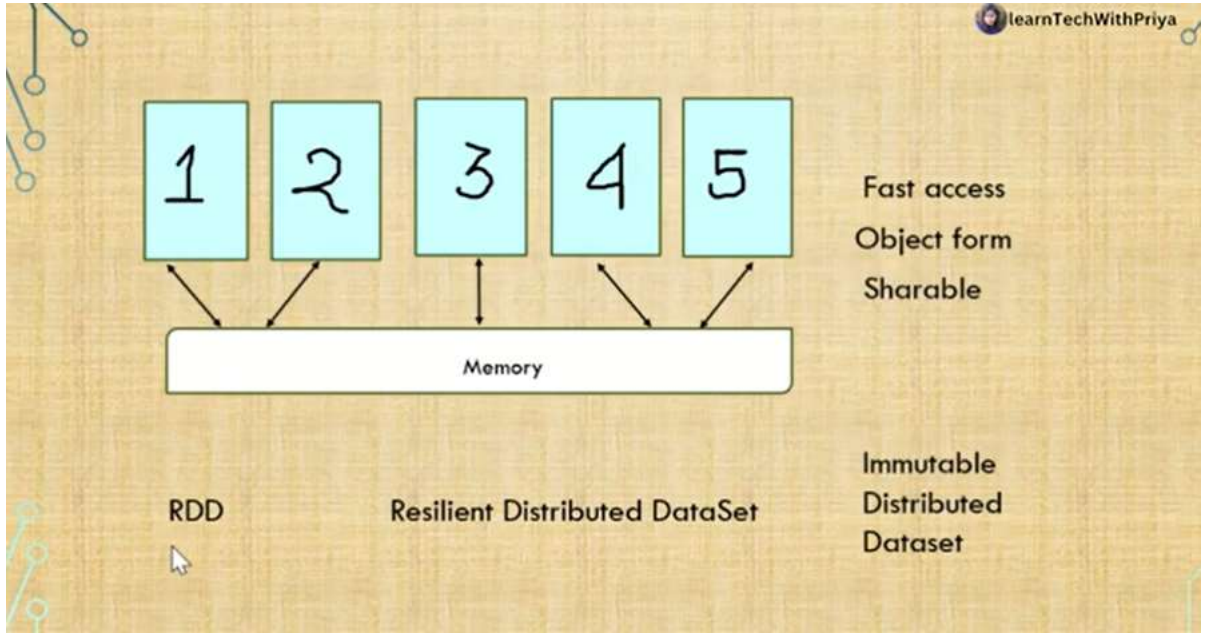
The Spark could be expanded as the stand-alone server. Also, it can be expanded on any shared computing framework such as YARN or Mesos.

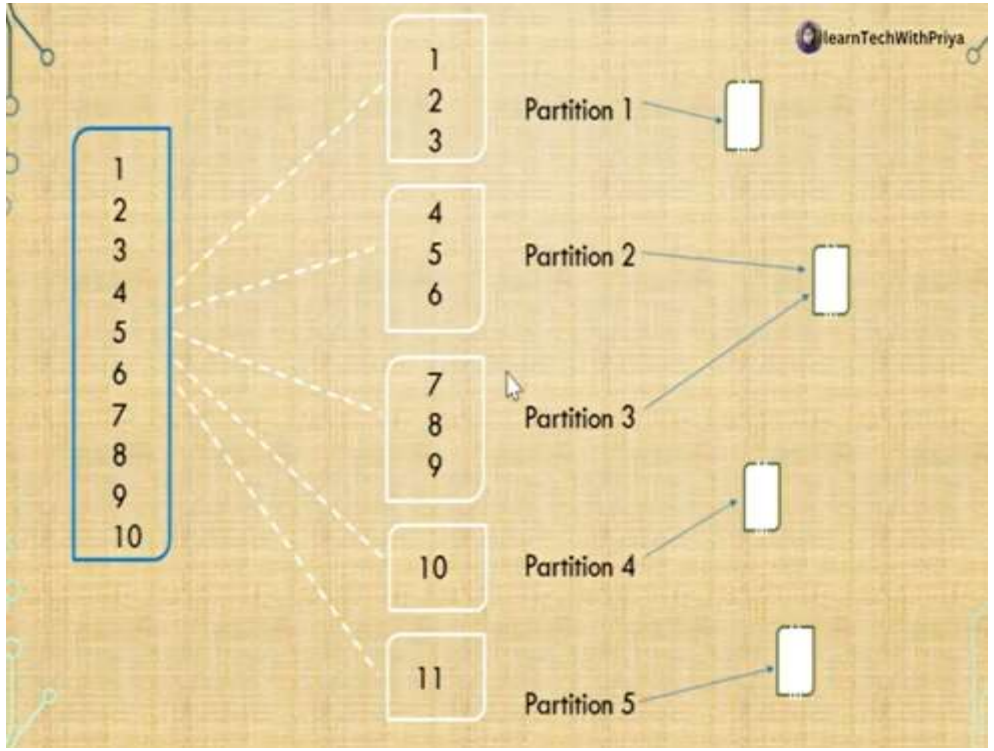
RDD IN SPARK

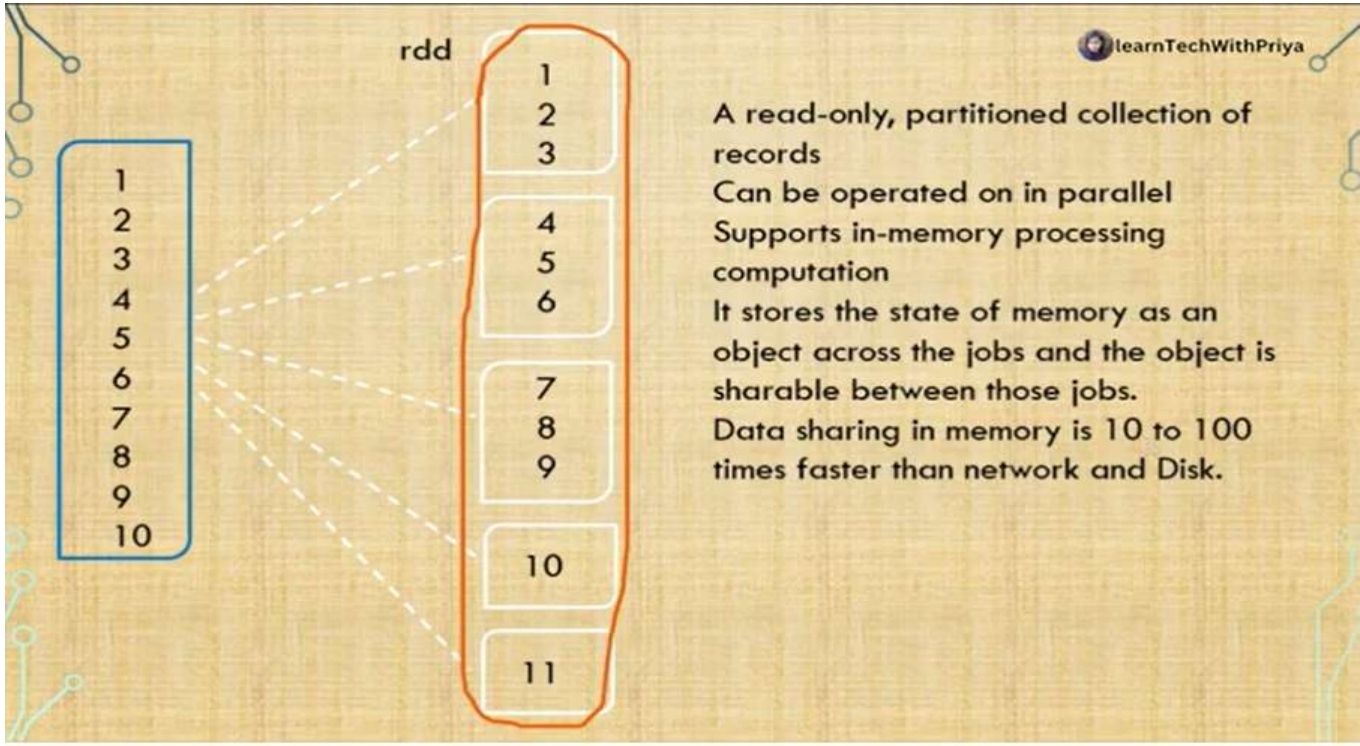
RDD stands for **Resilient Distributed Dataset**. It is a core concept within the Spark framework. Assume RDD like any table inside the database. It could take any data type. Spark can store data in Resilient Distributed Dataset on distinct partitions.
These datasets can support to rearrange the computations and upgrading the processing of data.
Also, these datasets are fault-tolerant due to the RDD which understands how to re-compute and recreate the datasets.
Resilient Distributed Datasets are immutable. We can change RDD using a transformation. However, this transformation will return us to a newer RDD while the actual RDD endures the same.
In more detail, RDD provides its support for two kinds of operations:

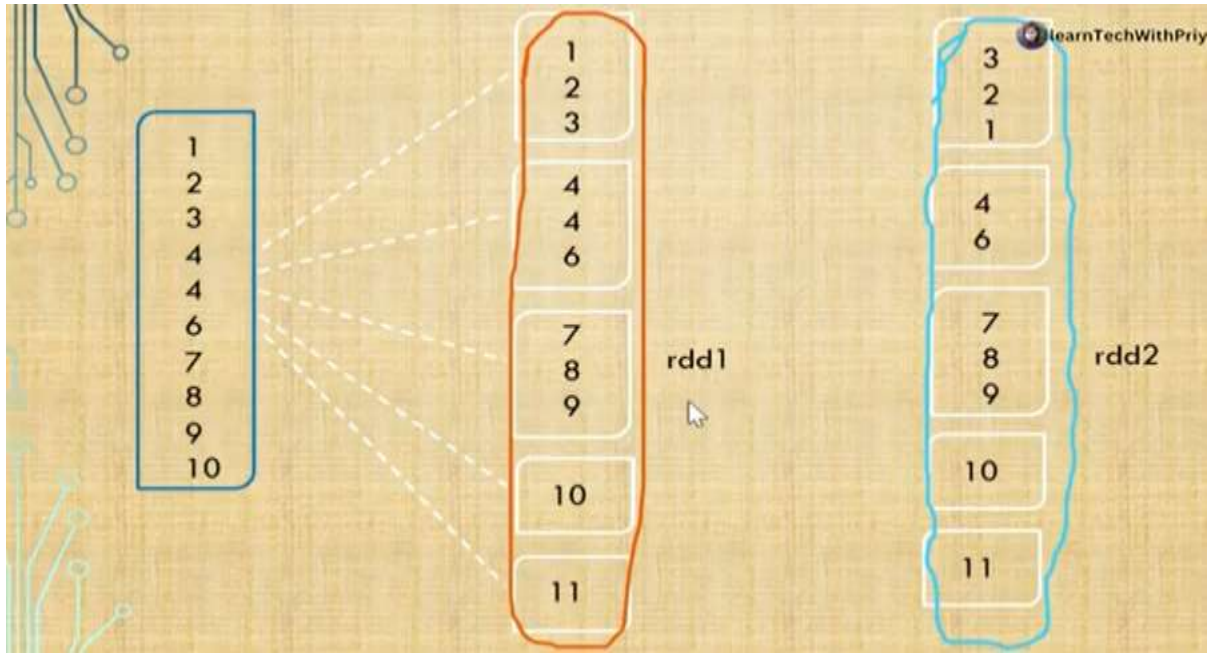

learnTechWithPriya
1 2 3 4 5 6 7 8 9 10
1 2 3
4 5 6
7 8 9
10
11
Partition 1
Partition 2
Partition 3
Partition 4
Partition 5
2GB
2GB
2GB
2GB


1 2 3 4 5
Disk I/O
Serialization
Replication


learnTechWithPriya
1 2 3 4 5
Memory
Fast access
Object form
Sharable
Immutable
Distributed
Dataset
RDD
Resilient Distributed DataSet


LearnTechWithPriya
1 2 3 4 5 6 7 8 9 10
1 2 3
4 5 6
7 8 9
10
11
Partition 1
Partition 2
Partition 3
Partition 4
Partition 5


LearnTechWithPriya
rdd
1 2 3 4 5 6 7 8 9 10
1 2 3
4 5 6
7 8 9
10
11
A read-only, partitioned collection of records
Can be operated on in parallel
Supports in-memory processing computation
It stores the state of memory as an object across the jobs and the object is sharable between those jobs.
Data sharing in memory is 10 to 100 times faster than network and Disk.


LearnTechWithPriy
1 2 3 4 4 6 6 7 8 9 10
1 2 3
4 4 6
7 8 9
10
11
rdd1
3 2 1
4 6
7 8 9
10
11
rdd2

RDD IN SPARK

1. **Action**
2. **Transformation**

- Action
- This operation assesses and returns a newer value. Each query of data processing is computed and the final value will be returned if a function of action is called over an RDD object.
  A few of the operations of Action are for each, count By Key, take, first, count, collect, and reduce.
- Transformation
- Transformation does not return any single value. It returns a **newer RDD**. Nothing will be assessed if we call any function of transformation. It only holds the RDD and then returns a newer RDD.

RDD IN SPARK

A few of the operations of Transformation are **coalesce, pipe, aggregateByKey, reduceByKey, groupByKey, flatMap, filter,** and **map.**

CONCEPT OF RESILIENT DISTRIBUTED DATASET (RDD)

RDD stands for Resilient Distribution Datasets. An RDD is a fault-tolerant collection of operational elements that run in parallel. The partitioned data in RDD is immutable and distributed in nature. There are primarily two types of RDD:

1. Parallelized Collections: Here, the existing RDDs running parallel with one another.
2. Hadoop Datasets: They perform functions on each file record in HDFS or other storage systems.

RDDs are basically parts of data that are stored in the memory distributed across many nodes. RDDs are lazily evaluated in Spark. This lazy evaluation is what contributes to Spark's speed.

HOW DO WE CREATE RDDS IN SPARK?

Spark provides two methods to create RDD:

1. By parallelizing a collection in your Driver program.

2. This makes use of SparkContext's 'parallelize'

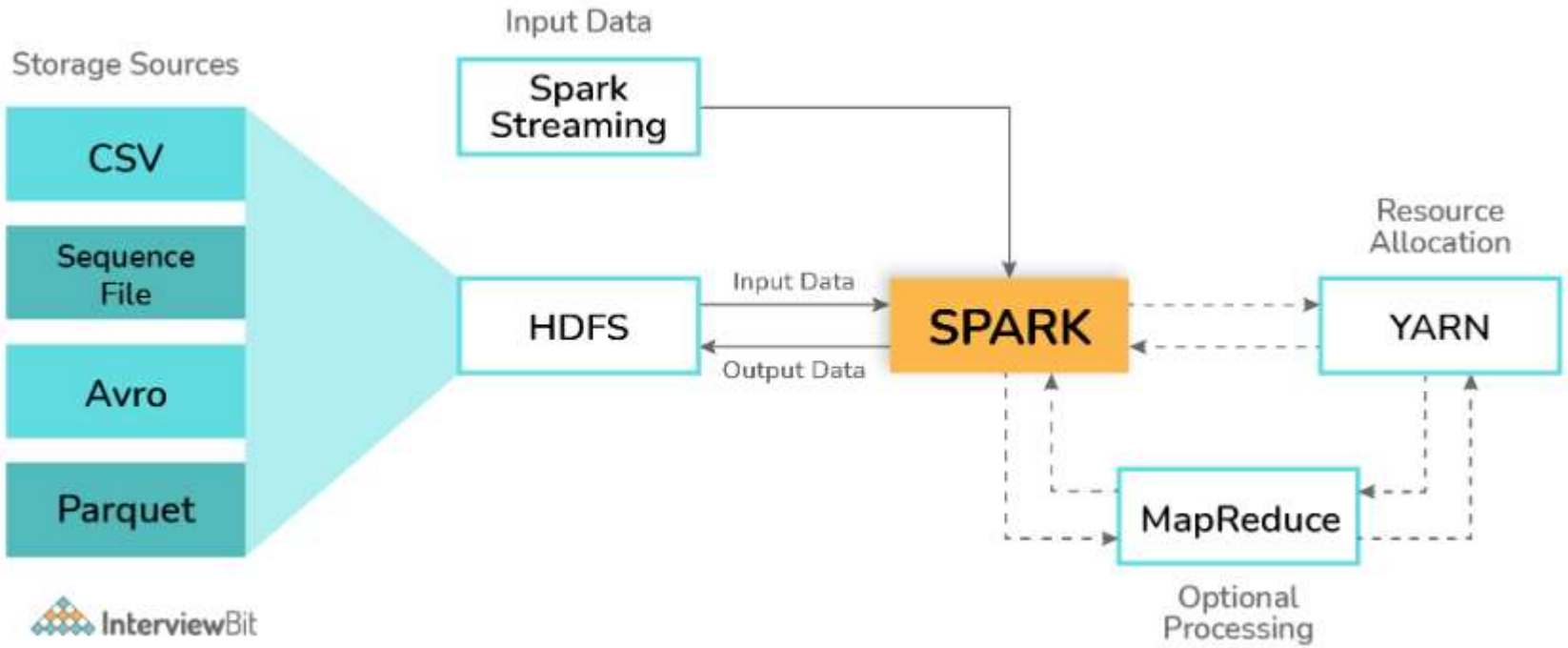
```
method val DataArray = Array(2,4,6,8,10)

val DataRDD = sc.parallelize(DataArray)
```

3. By loading an external dataset from external storage like HDFS, HBase, shared file system.

CAN APACHE SPARK BE USED ALONG WITH HADOOP? IF YES, THEN HOW?

Yes! The main feature of Spark is its compatibility with Hadoop. This makes it a powerful framework as using the combination of these two helps to leverage the processing capacity of Spark by making use of the best of Hadoop's YARN and HDFS features.



CAN APACHE SPARK BE USED ALONG WITH HADOOP? IF YES, THEN HOW?

Hadoop can be integrated with Spark in the following ways:

HDFS: Spark can be configured to run atop HDFS to leverage the feature of distributed replicated storage.

MapReduce: Spark can also be configured to run alongside the MapReduce in the same or different processing framework or Hadoop cluster. Spark and MapReduce can be used together to perform real-time and batch processing respectively.

YARN: Spark applications can be configured to run on YARN which acts as the cluster management framework

SPARK'S GRAPHX

Spark's GraphX is a distributed graph processing framework that provides a high-level API for performing graph computation on large-scale graphs. GraphX allows users to express graph computation as a series of transformations and provides optimized graph processing algorithms for various graph computations such as PageRank and Connected Components.

HOW DOES IT DIFFER FROM OTHER GRAPH PROCESSING FRAMEWORKS

Compared to other graph processing frameworks such as Apache Graph and Apache Flink, GraphX is tightly integrated with Spark and allows users to combine graph computation with other Spark features such as machine learning and streaming. GraphX provides a more concise API and better performance for iterative graph computations.

VARIOUS KINDS OF OPERATORS PROVIDED BY SPARK GRAPHX

Apache Spark GraphX provides threse types of operators which are:

- Property operators: Property operators produce a new graph by modifying the vertex or edge properties using a user-defined map function. Property operators usually initialize a graph for further computation or remove unnecessary properties.
- Structural operators: Structural operators work on creating new graphs after making structural changes to existing graphs.
- The reverse method returns a new graph with the edge directions reversed.

VARIOUS KINDS OF OPERATORS PROVIDED BY SPARK GRAPHX

- The subgraph operator takes vertex predicates and edge predicates as input and returns a graph containing only vertices that satisfy the vertex predicate and edges satisfying the edge predicates and then connects these edges only to vertices where the vertex predicate evaluates to "true."
- The mask operator is used to construct a subgraph of the vertices and edges found in the input graph.
- The groupEdges method is used to merge parallel edges in the multigraph. Parallel edges are duplicate edges between pairs of vertices.
- Join operators: Join operators are used to creating new graphs by adding data from external collections such as resilient distribution datasets to charts.