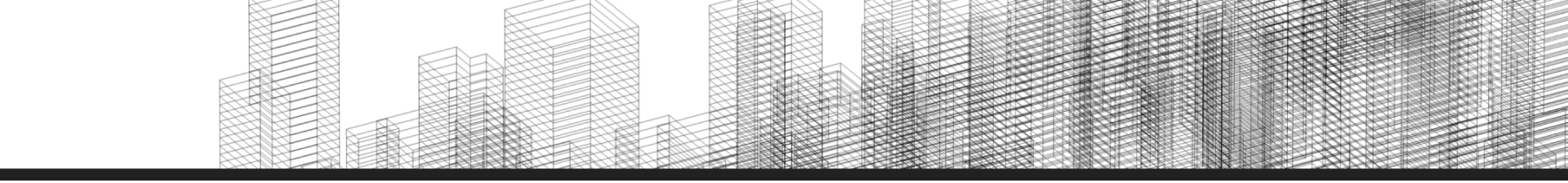
APACHE SPARK

An Introduction



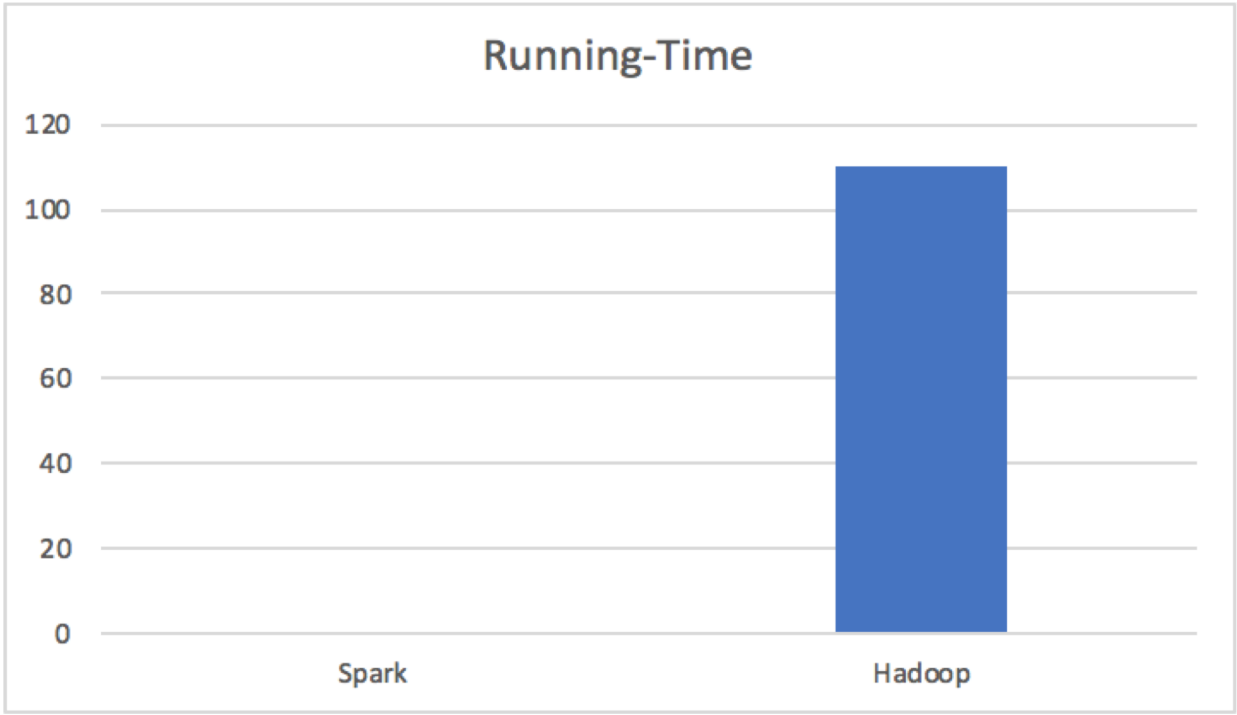
What is Apache Spark?

Apache Spark is a fast and multi-purpose engine for large-scale data processing.

THERE ARE FOUR REASONS TO USE SPARK

1. Speed
2. Ease of use
3. Generality
4. Platform-agnostic

SPEED



EASE OF USE

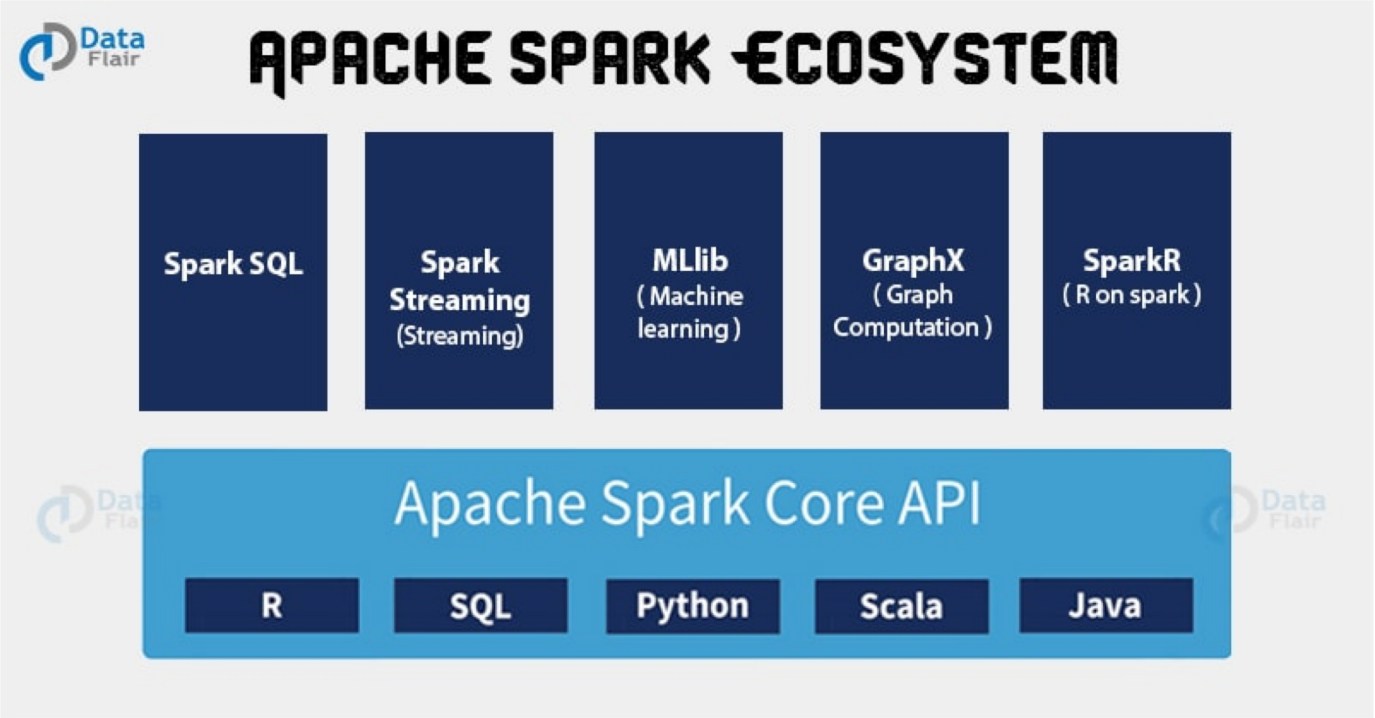
1 Spark supports Java, Scala, Python, and R natively, as well as ANSI SQL.

2 It offers 80 high-level operators making it fast and easy to build applications, even including parallelization and streaming.

3 We can use popular interfaces like Jupyter Notebook, Apache Zeppelin, as well as the command shell.

4 By using just two lines of code, you can count all words in a large file.

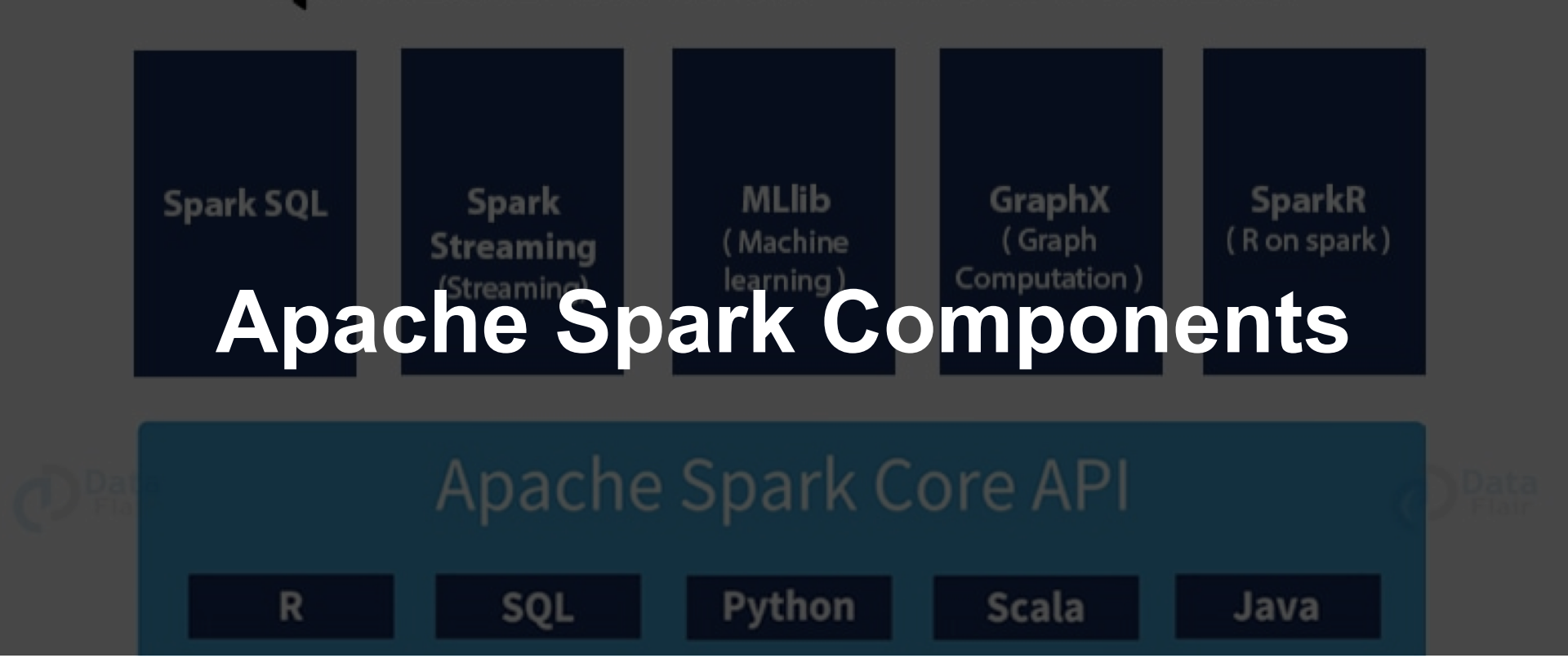
GENERALITY


Data Flair
APACHE SPARK ECOSYSTEM
Spark SQL
Spark Streaming (Streaming)
MLlib (Machine learning)
GraphX (Graph Computation)
SparkR (R on spark)
Apache Spark Core API
R
SQL
Python
Scala
Java

PLATFORM AGNOSTIC

Besides running in nearly any environment you can access the data in the Hadoop distributed file system, known as HDFS, Cassandra, HBASE, Hive, or any other Hadoop data source.

In Spark 2.0 you can also connect directly to traditional relational databases using dataframes in Python and Scala.



Apache Spark Components

How Spark's components fit together?

SPARK CORE

- Fundamental Component
- Task distribution
- Scheduling
- Input/Output operations

SPARK SQL

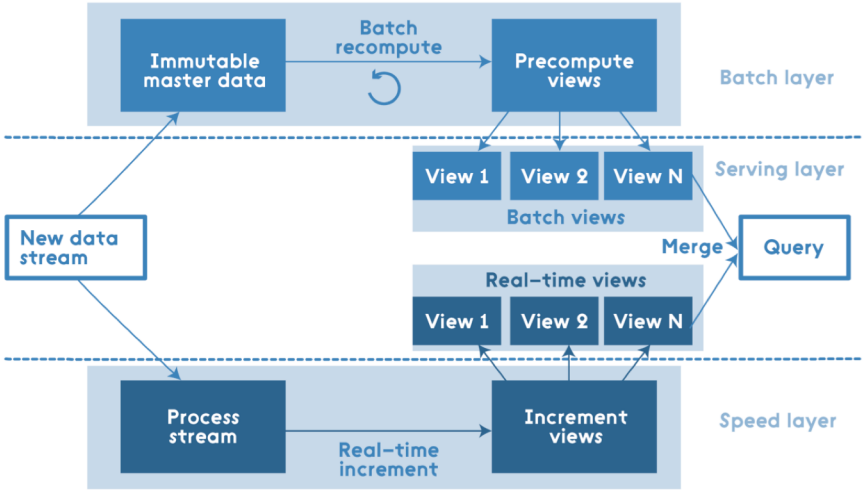
- It supports the ANSI SQL
- Enable tools like Tableau to easily integrate with Spark
- DataFrames
    - Spark SQL provides Dataframe concept that is a familiar term for data science.

SPARK Streaming

Streaming Analytics

Micro Batches

Lambda Architecture



SPARK MLLib

- The MLlib component enables machine learning algorithms to run.
- It is 9X faster than Apache Mahout.
- It includes common functions

SPARK GRAPHX

- Graph Processing
- It is in-memory version of Apache Giraph.
- Based on RDDs

APACHE GIRAPH

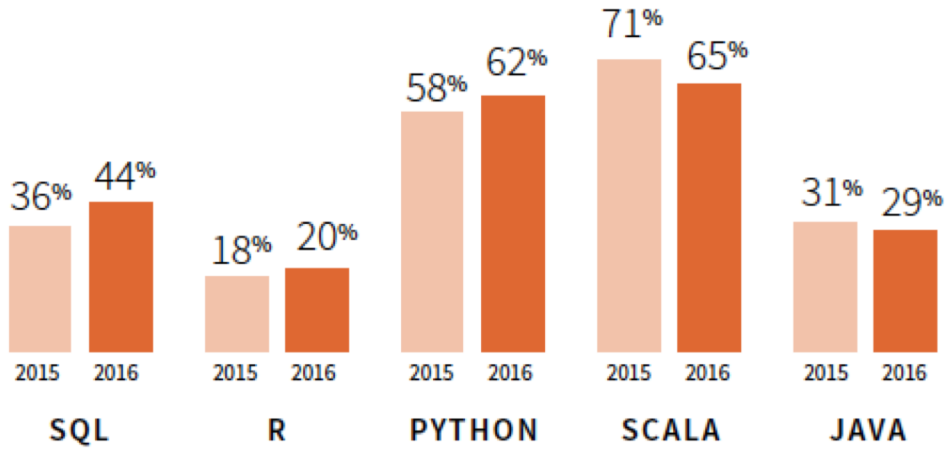
SPARK R

- R package for Spark
- It provides an interface for connecting your Spark cluster from the R statistical package.
- This package provides Distributed DataFrames, which are comparable to DataFrames in R.
- R Studio integration

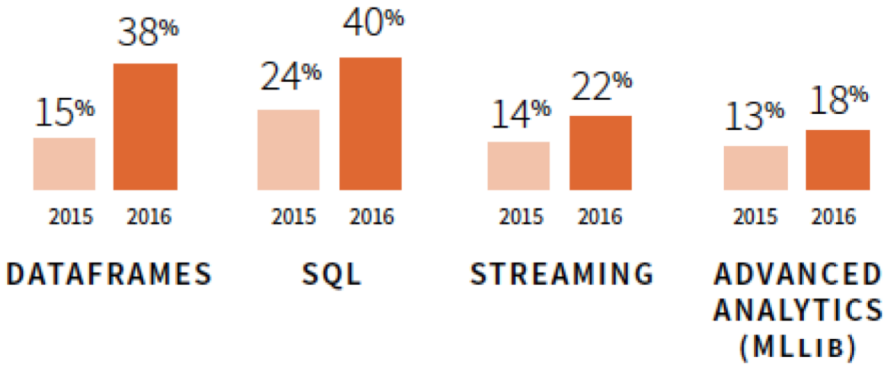
The usage of Apache Spark

- Data integration or ETL
- Machine Learning
- BI/Analytics
- Real-Time Processing
- Recommendation Engines

Languages used in Spark



Spark components used in production

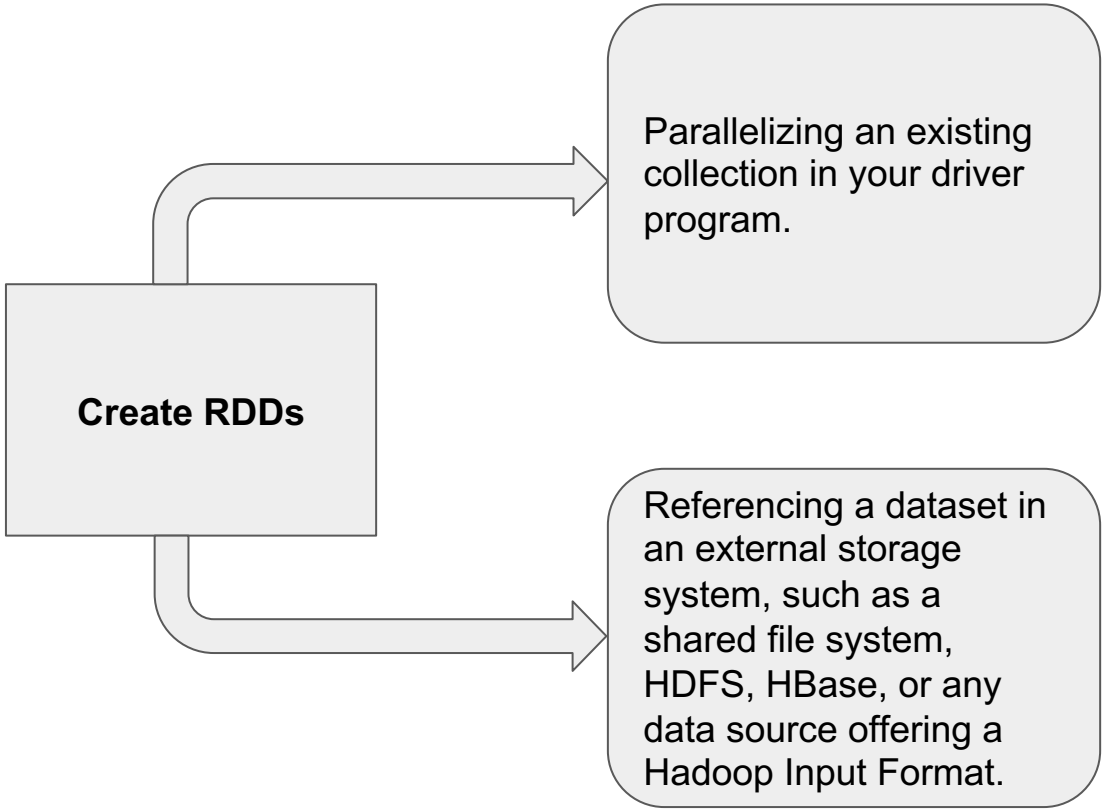


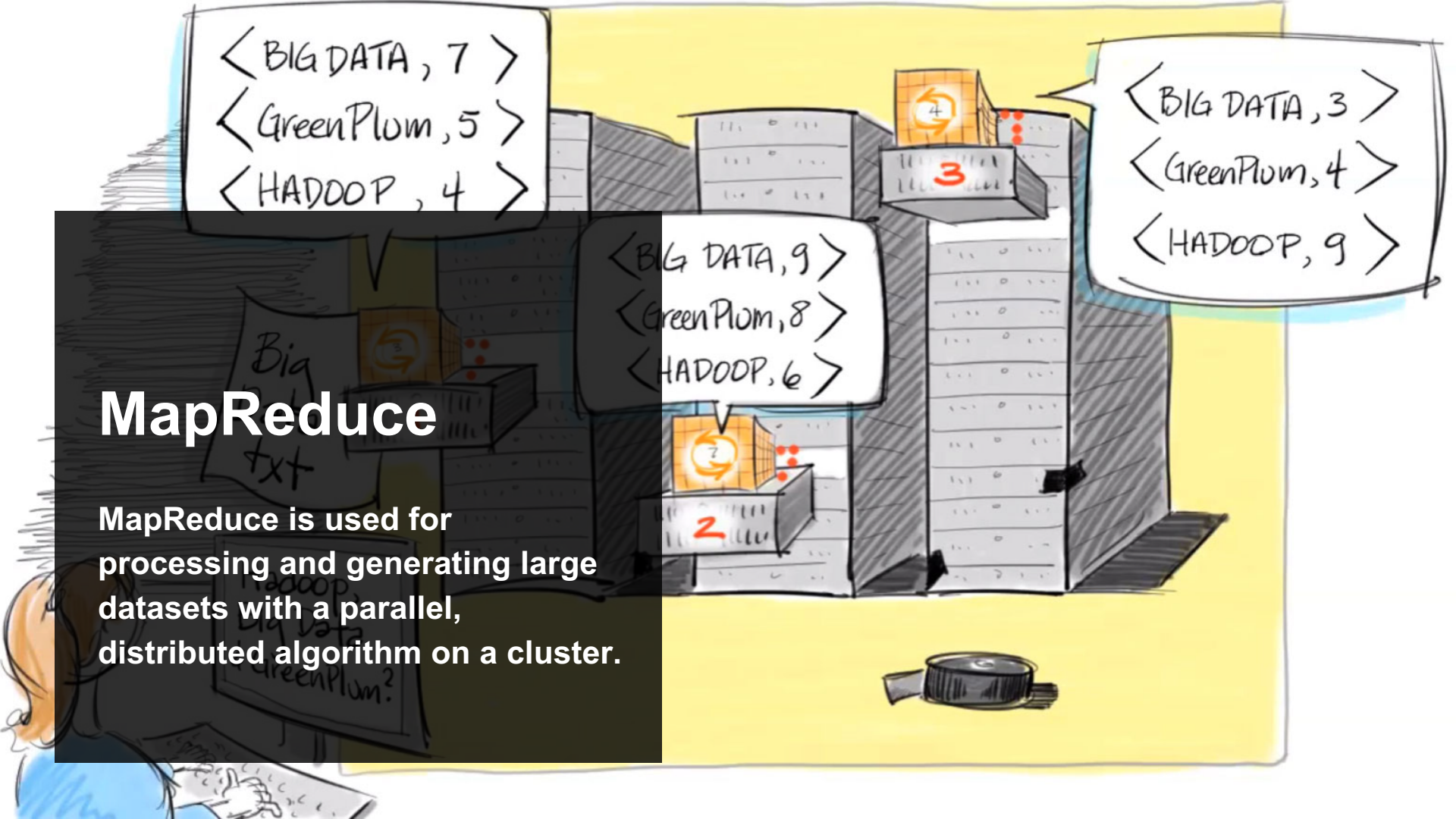
Deep Dive into Apache Spark

Resilient Distributed Datasets (RDD)

- RDD is a fundamental data structure of Spark.
- It is an immutable distributed collection of objects.
- Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster.
- RDD can contain any type of objects, including user-defined classes.
- RDD is a fault-tolerant collection of elements that can be operated on parallelly.

How to Create RDDs?





MapReduce

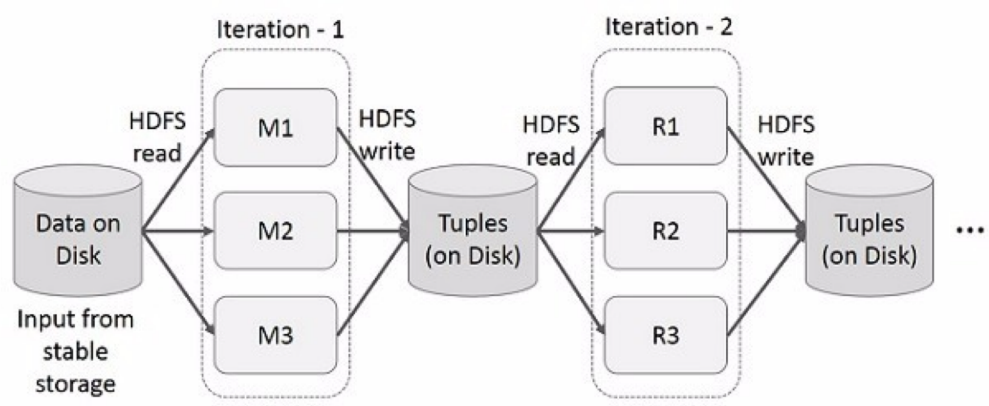
MapReduce is used for processing and generating large datasets with a parallel, distributed algorithm on a cluster.

Data sharing in MapReduce

Data sharing is slow in **MapReduce** due to **replication**, **serialization**, and **disk IO**. Regarding storage system, most of the Hadoop applications, they spend more than 90% of the time doing HDFS read-write operations.

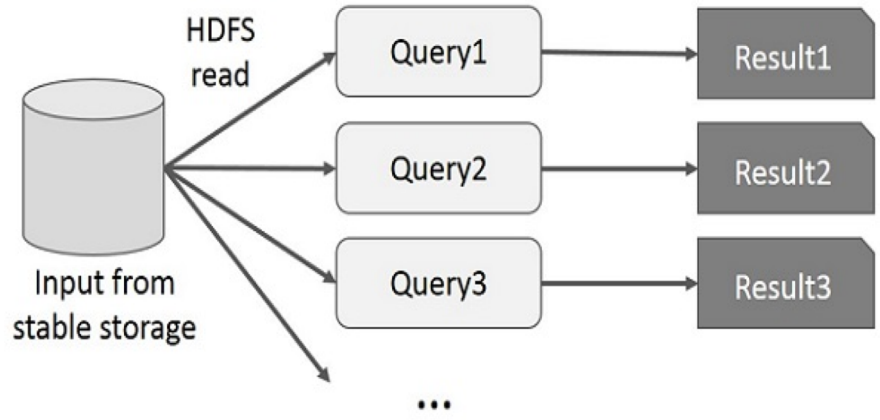
Iterative Operations on MapReduce

- Reuse intermediate results across multiple computations in multi-stage applications.
- The following illustration explains how the current framework works, while doing the iterative operations on MapReduce.
- This incurs substantial overheads due to data replication, disk I/O, and serialization, which makes the system slow.



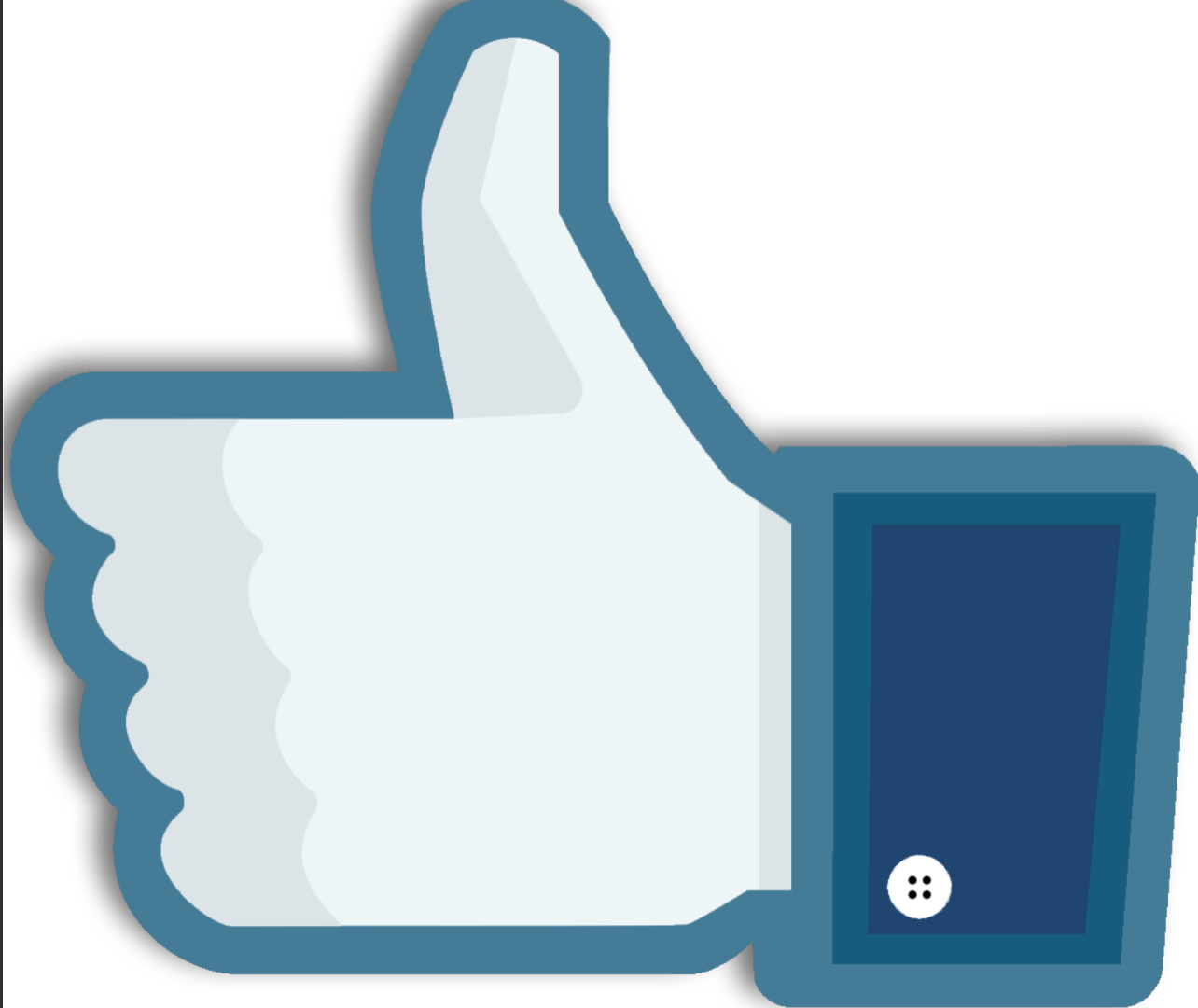
Interactive Operations on MapReduce

- Each query will do the disk I/O on the stable storage, which can dominate application execution time.
- The following illustration explains how the current framework works while doing the interactive queries on MapReduce.



Data Sharing using Spark RDD

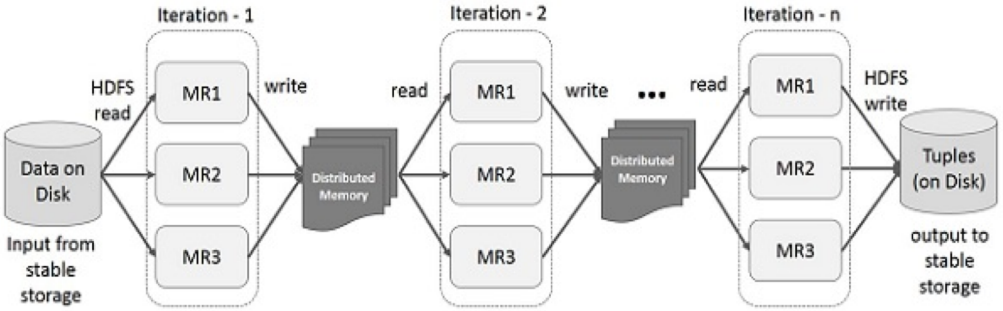
- RDD supports in-memory processing computation. it stores the state of memory as an object across the jobs and the object is shareable between those jobs.
- Data sharing in memory is 10 to 100 times faster than network and disk.



Iterative Operations on Spark RDD

The illustration given below shows the iterative operations on Spark RDD. It will store intermediate results in a distributed memory instead of Stable storage (Disk) and make the system faster.

If the Distributed memory (RAM) is not sufficient to store intermediate results (State of the JOB), then it will store those results on the disk.



Interactive Operations on Spark RDD

- The following illustration shows interactive operations on Spark RDD.
- If different queries are run on the same set of data repeatedly, this particular data can be kept in memory for better execution times.
- By default, each transformed RDD may be recomputed each time you run an action on it.
- However, you may also persist an RDD in memory, in which case Spark will keep the elements around on the cluster for much faster access, the next time you query it.
- There is also support for persisting RDDs on disk, or replicated across multiple nodes.