Big Data Hadoop and Spark Developer

Processing RDDs

Learning Objectives

By the end of this lesson, you will be able to:

- Define Spark RDD and list its limitations
- Describe and demonstrate RDD operations in Spark
- Demonstrate the creation of Spark RDD
- Aggregate data with pair RDD

Introduction to Spark RDD

Spark RDD

Spark Resilient Distributed Dataset (RDD) is an immutable collection of objects which defines the data structure of Spark.

The following are the features of Spark RDD:

1. Lazy Evaluation
2. Coarse-Grained Operation
3. In-Memory Computation
4. Fault-Tolerant
5. Partitioning
6. Persistent
7. Immutable
8. Location-Stickiness

RDD in Spark

The following are the reasons for the evolution of RDD:

1. Iterative algorithm
2. Interactive data mining tools
3. Inefficient implementation of Distributed Shared Memory (DSM)
4. Slower computation when distributed computing systems store data in HDFS or Amazon S3

Limitations of Spark RDD

Data Types Supported by RDD

Primitive

- Integer
- Character
- Boolean

Sequence

- Strings
- Lists
- Arrays
- Tuples
- Dicts
- Nested

Java and Scala objects

Mixed Data

Creating Spark RDD

Creating Spark RDD

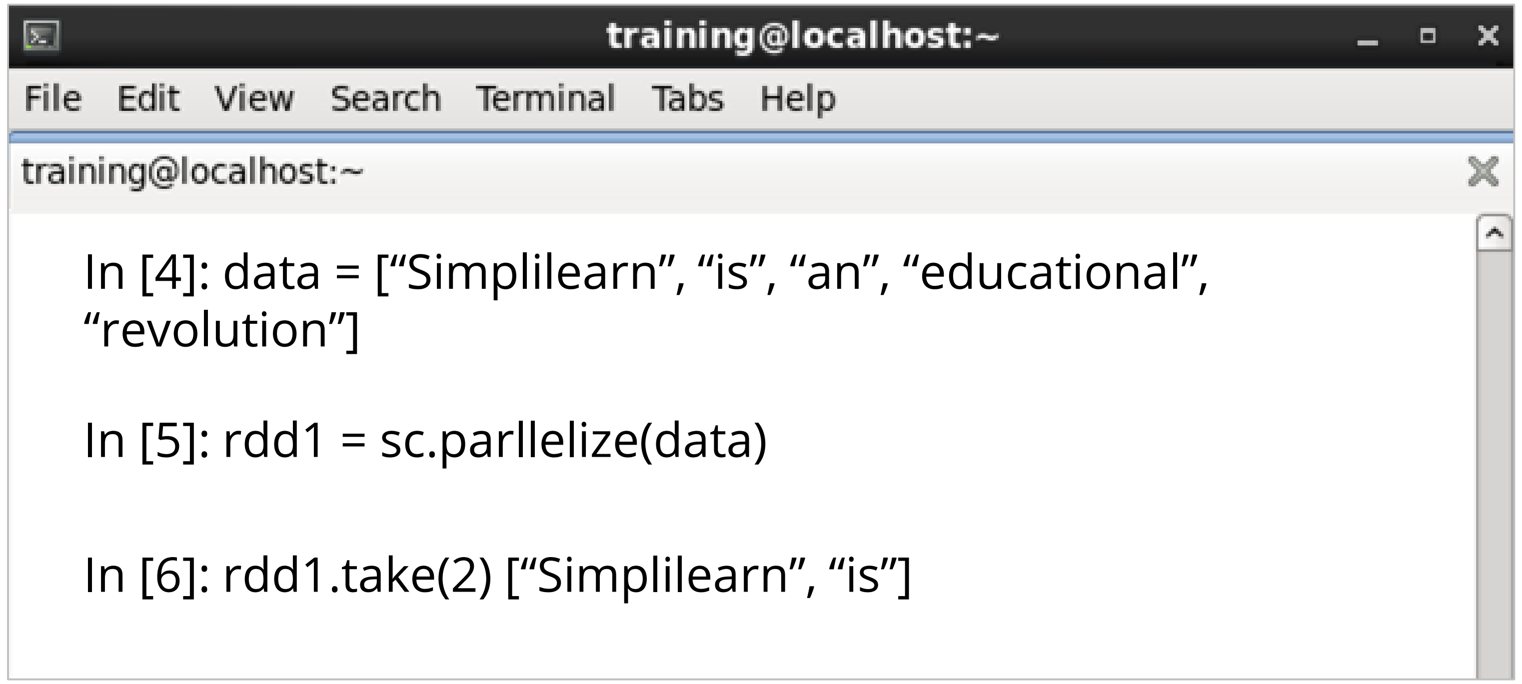
Parallelized Collections

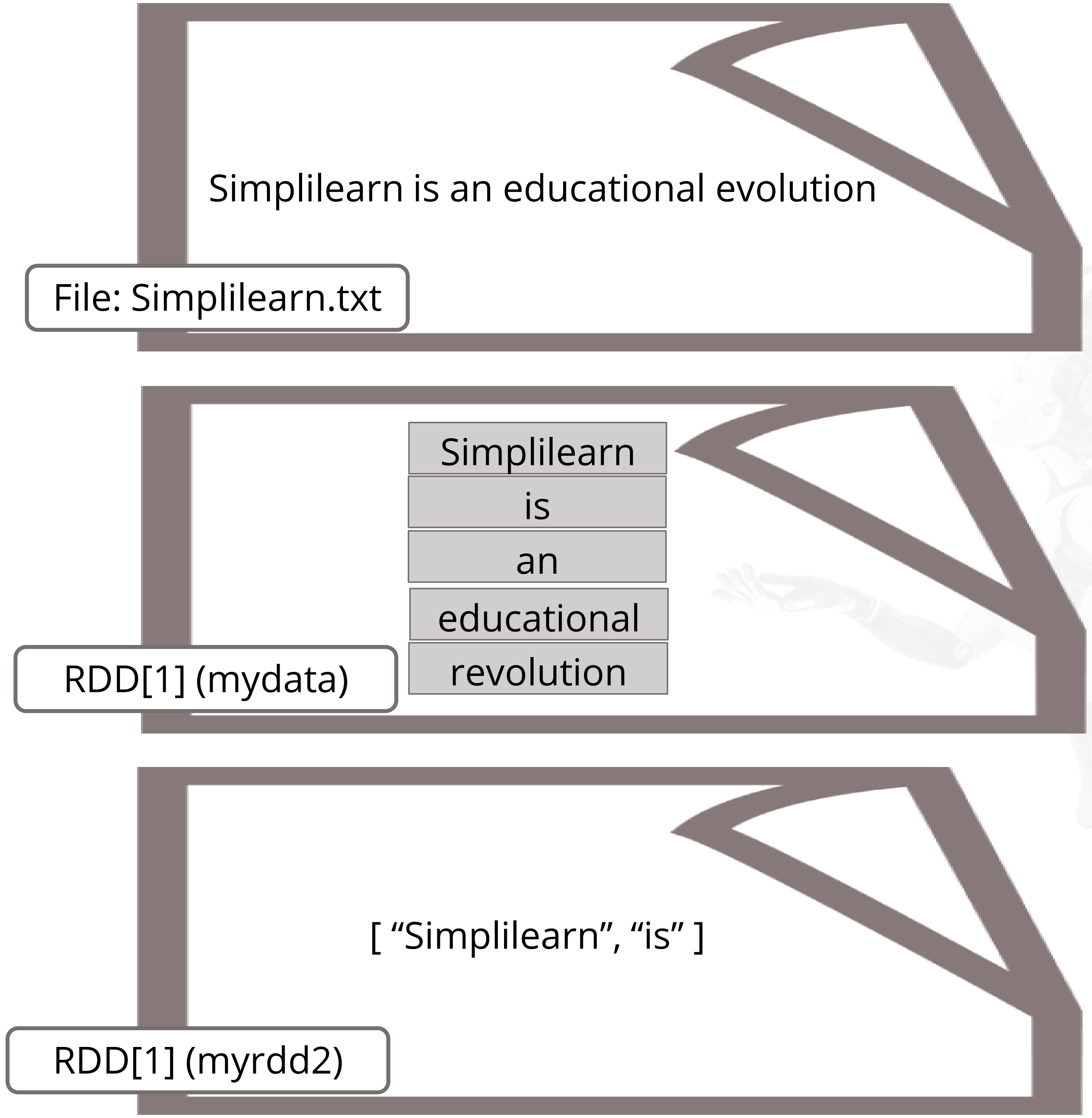
RDDs are created by parallelizing an existing collection in your driver program or referencing a dataset in an external storage system.

RDDs are created by taking the existing collection and passing it to SparkContext parallelize() method.

```
val data=spark.sparkContext.parallelize(Seq((“physics",78),(“chemistry",48),(“biology",73),
(“english",54),("maths",77)))
val sorted = data.sortByKey()
sorted.foreach(println)
```

Creating RDD from Collections





Existing RDDs

RDDs can be created from existing RDDs by transforming one RDD into another RDD.

```
val words=spark.sparkContext.parallelize(Seq("Simplilearn", "is", "an", "education", "provider"))

val wordPair = words.map(w => (w.charAt(0), w))

wordPair.foreach(println)
```

External Data

In Spark, a dataset can be created from any other dataset. The other dataset must be supported by Hadoop, including the local file system, HDFS, Cassandra, HBase, and many more.

Data frame reader interface can be used to load dataset from an external storage system in the following formats:

CSV	`val dataRDD = spark.read.csv("path/of/csv/file").rdd`
JSON	`val dataRDD = spark.read.json("path/of/json/file").rdd`
Textfile	`val dataRDD = spark.read.textFile("path/of/text/file").rdd`

Creating RDD from a Text File

To create a file-based RDD, you can use the command SparkContext.textFile or sc.textfile, and pass one or more file names.

Text File

Data in Memory

```
training@localhost:~
File  Edit  View  Search  Terminal  Tabs  Help
training@localhost:~

    sc.textFile(“simplilearn/*.log")
    sc.textFile(“simplilearn.txt)
    sc.textFile(“simplilearn1.txt,simplilearn2.txt")
```

Creating RDD from a Text File

Simplilearn is an education provider.\n

It is based in San Francisco. \n

It has trained 450,000+ customers. \n

It offers 400+ professional courses. \n

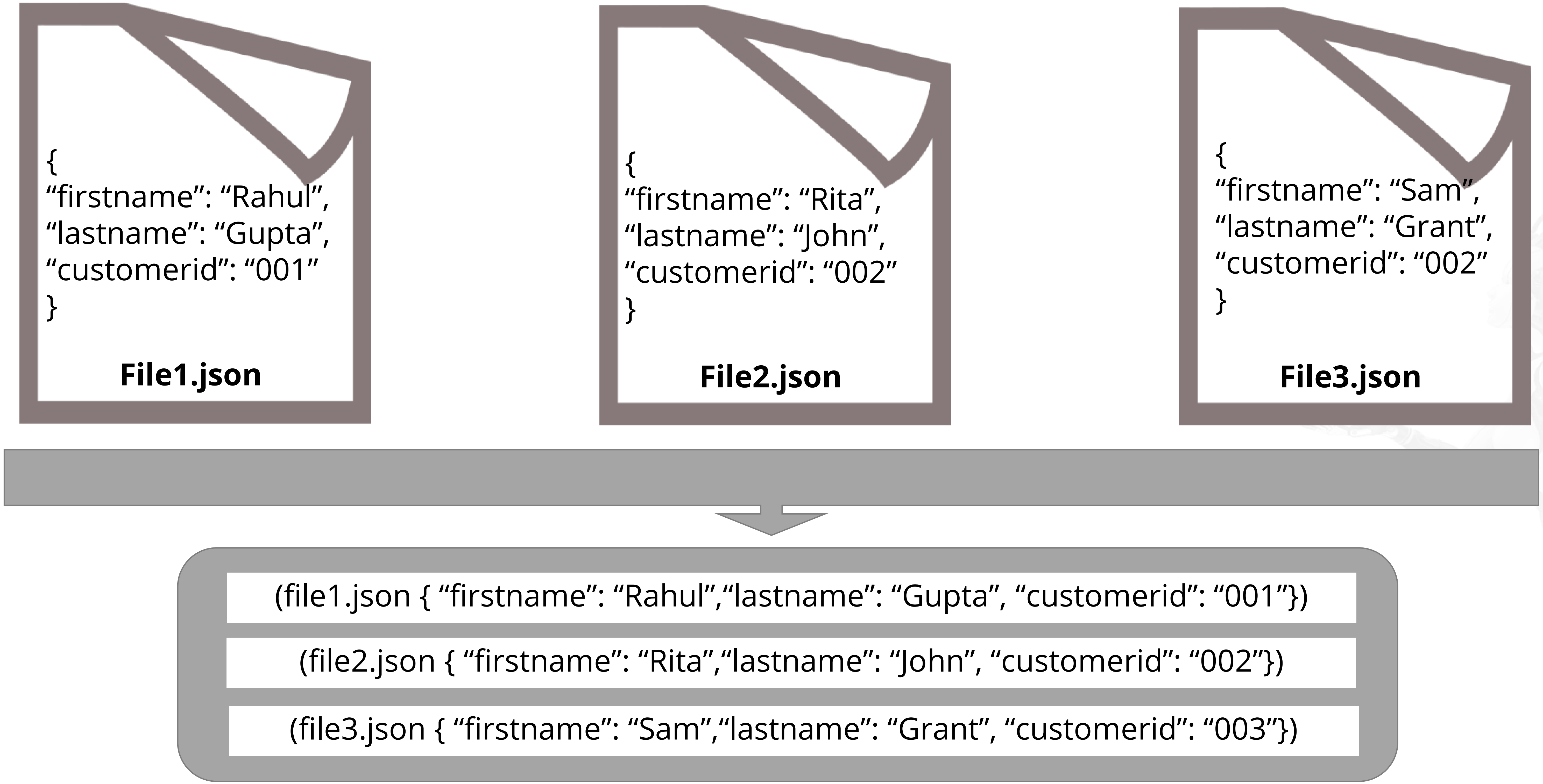
Simplilearn is an education provider.

It is based in San Francisco.

It has trained 450,000+ customers.

It offers 400+ professional courses.

Creating RDD from a Text File



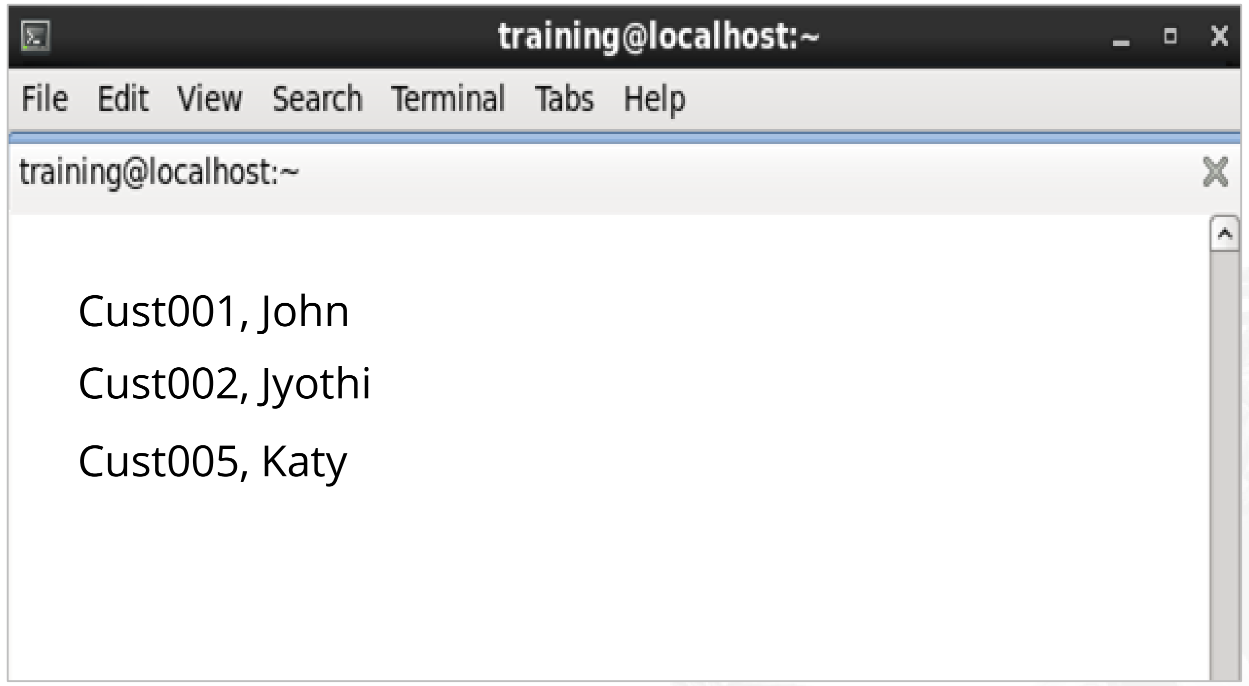
DATA AND ARTIFICIAL INTELLIGENCE

Pair RDD

Pair RDD and Double RDD

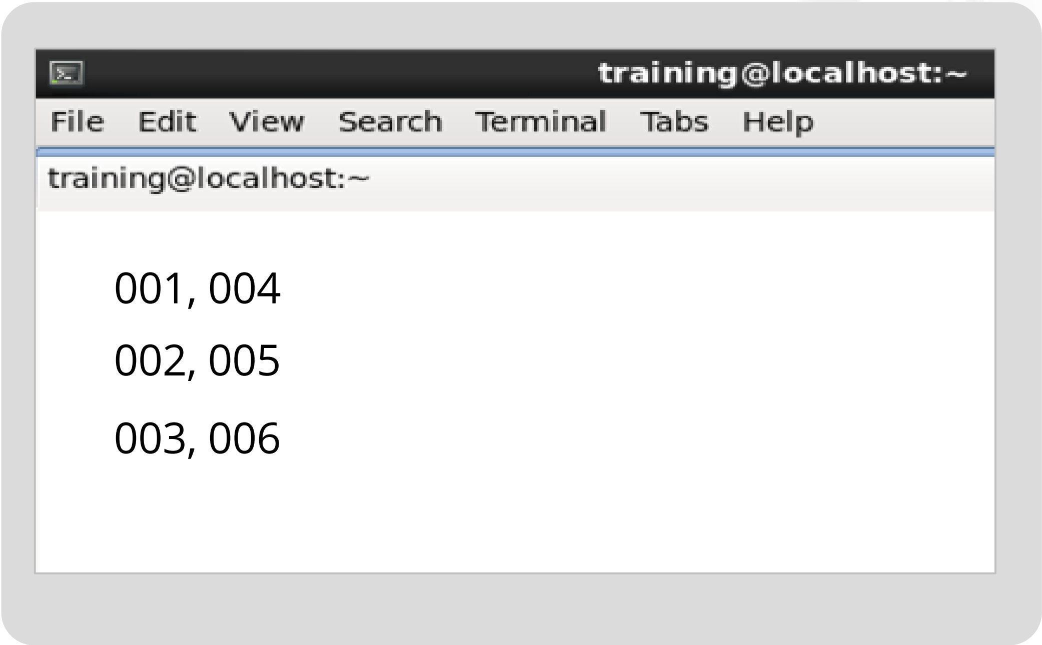
Pair RDDs

Some of the functions that can be performed with Pair RDDs are Map and flatMap.



Double RDDs

Double RDDs are RDDs that hold numerical data. Some of the functions that can be performed with Double RDDs are Distinct and Sum.



Creating Pair RDD

To create a Pair RDD, use functions such as Map, flatMap or flatMapValues, and keyBy.

Cust001 \t Deepak Mehta
Cust002 \t Seema Arora
Cust003 \t Asha Rao

Language: Python

```
Users = sc.textFile(file) \
.map (lambda line: line.split ('\t')) \
 .map (lambda fields: (fields [0], fields [1]))
```

Language: Scala

```
Val users = sc.textFile(file) .
    .map (line => line.split ('\t').
.map (fields => (fields (0), fields (1) ))
```

(Cust001, Deepak Mehta)

(Cust002, Seema Arora)

(Cust003, Asha Rao)

Pair RDD

DATA AND
ARTIFICIAL INTELLIGENCE

RDD Operations

RDD Operations

RDD supports the following types of operations:

Transformation

Action

Transformation

Transformation allows us to create new dataset using the existing one.

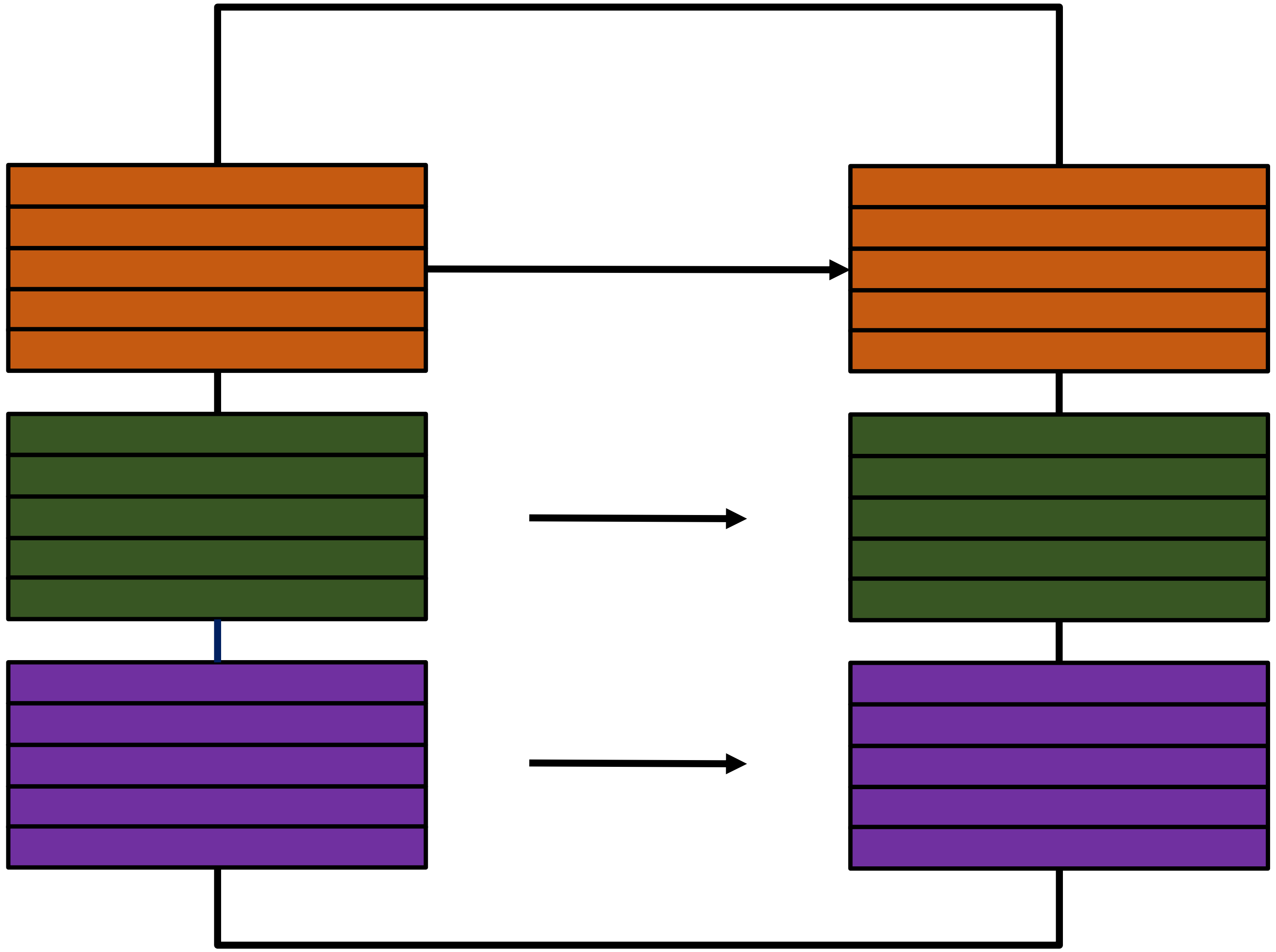
There are two types of transformation:

Narrow Transformation

Wide Transformation

Narrow Transformation

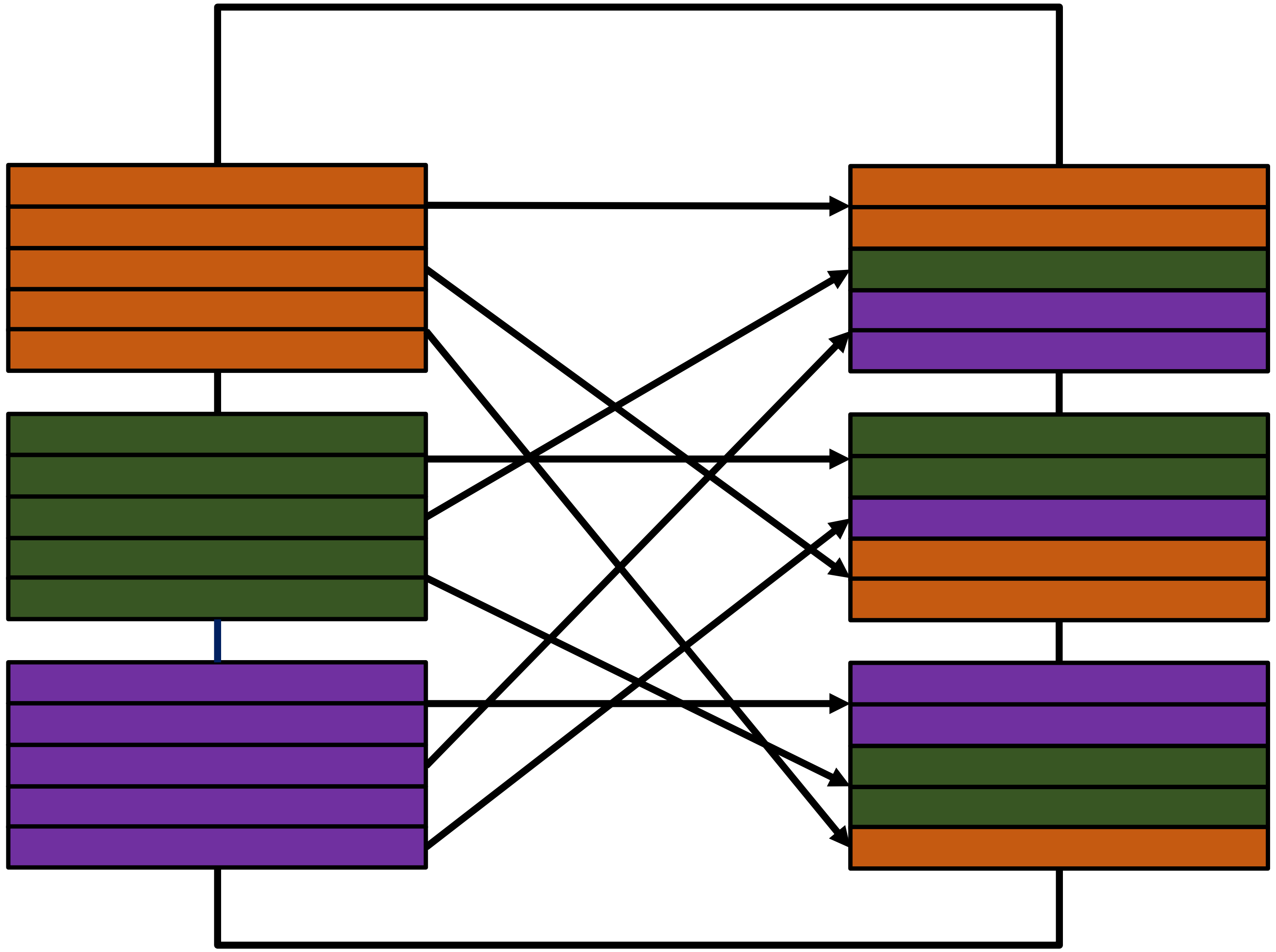
Narrow transformation is self-sufficient. It is the result of map and filter, such that the data is from a single partition only.



- Map
- Filter
- FlatMap
- Sample
- MapPartition
- Union

Wide Transformation

Wide transformation is not self-sufficient. It is the result of GroupByKey() and ReduceByKey() like functions, such that the data can be from multiple partitions.



- Cartesian
- ReduceByKey
- Join
- GroupByKey
- Intersection
- Coalesce
- Repartition
- Distinct

Spark Transformation: Detailed Exploration

Duration: 10 mins

Problem Statement: In this demonstration, you will explore Spark transformation.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Action

Action allows us to return a value to the driver program, after running a computation on the dataset.
Actions are the RDD operations that produce non-RDD values.

Reduce — Reduce is an action that aggregates all the elements of the RDD using some function.

- **First()**
- **Take()**
- **Reduce()**
- **Collect()**
- **Count()**

Spark Action: Detailed Exploration

Duration: 10 mins

Problem Statement: In this demonstration, you will explore Spark action.

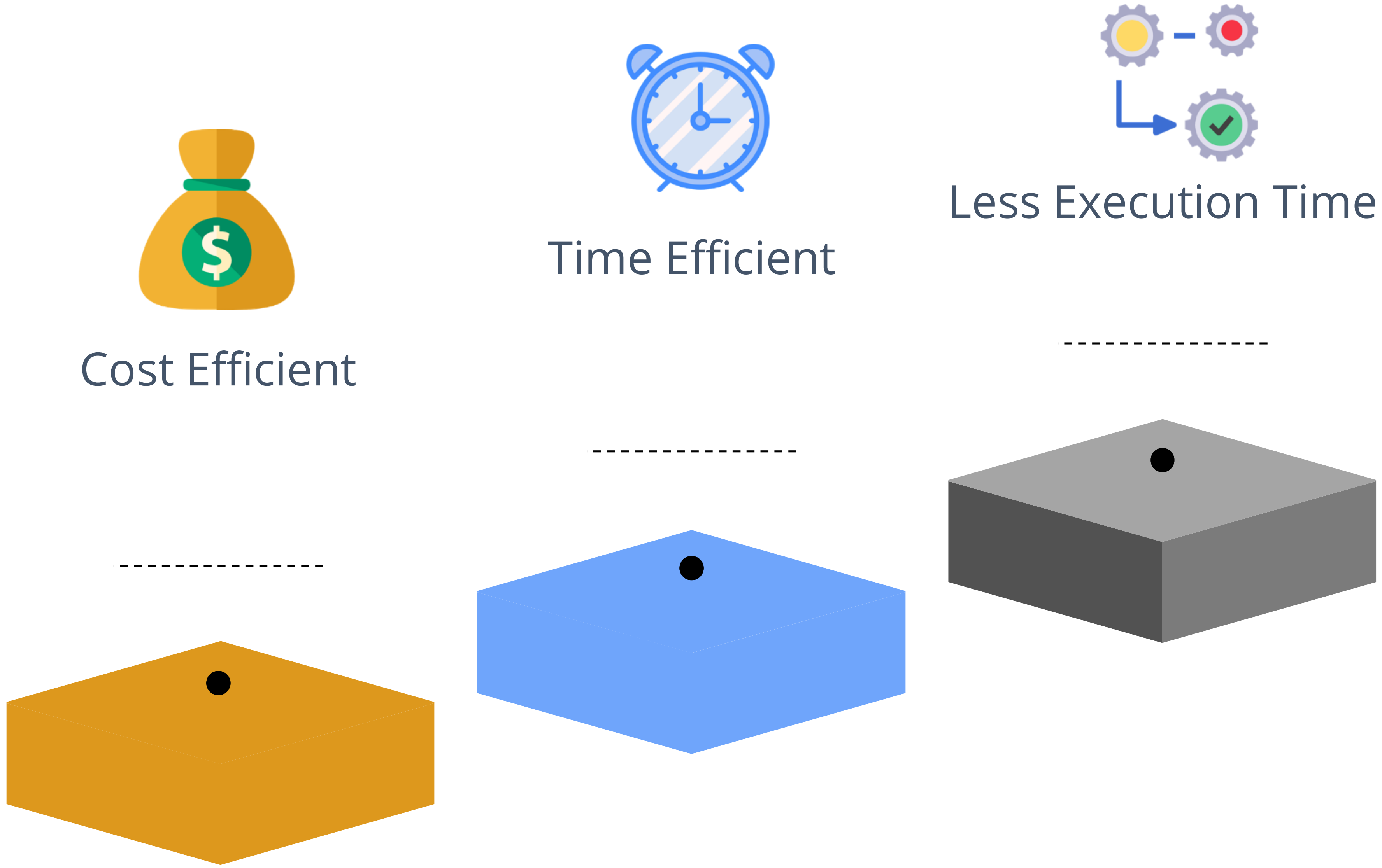
Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Caching and Persistence

Caching and Persistence

Caching and Persistence are the techniques used to store the result of RDD evaluation. They are also used by developers to enhance the performance of applications.

The following are the benefits of Caching and Persistence:



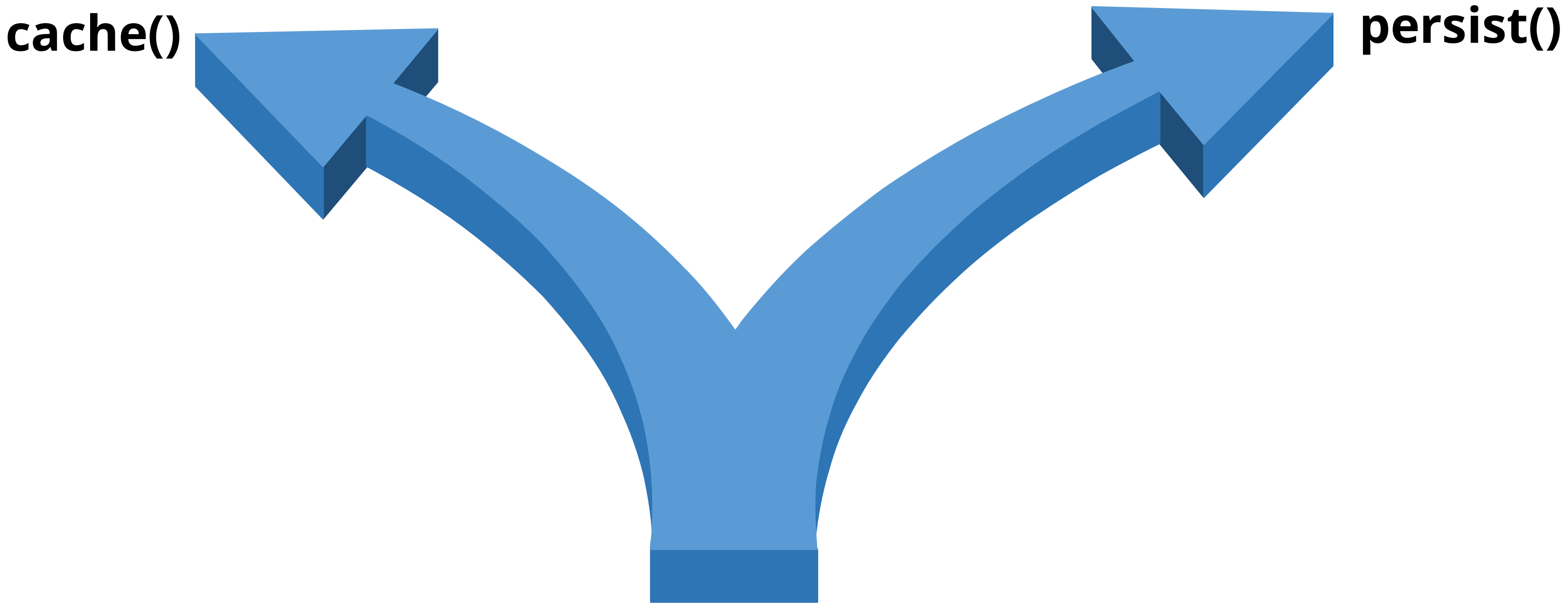
Features of RDD Persistence

Storage and reuse of the RDD partitions

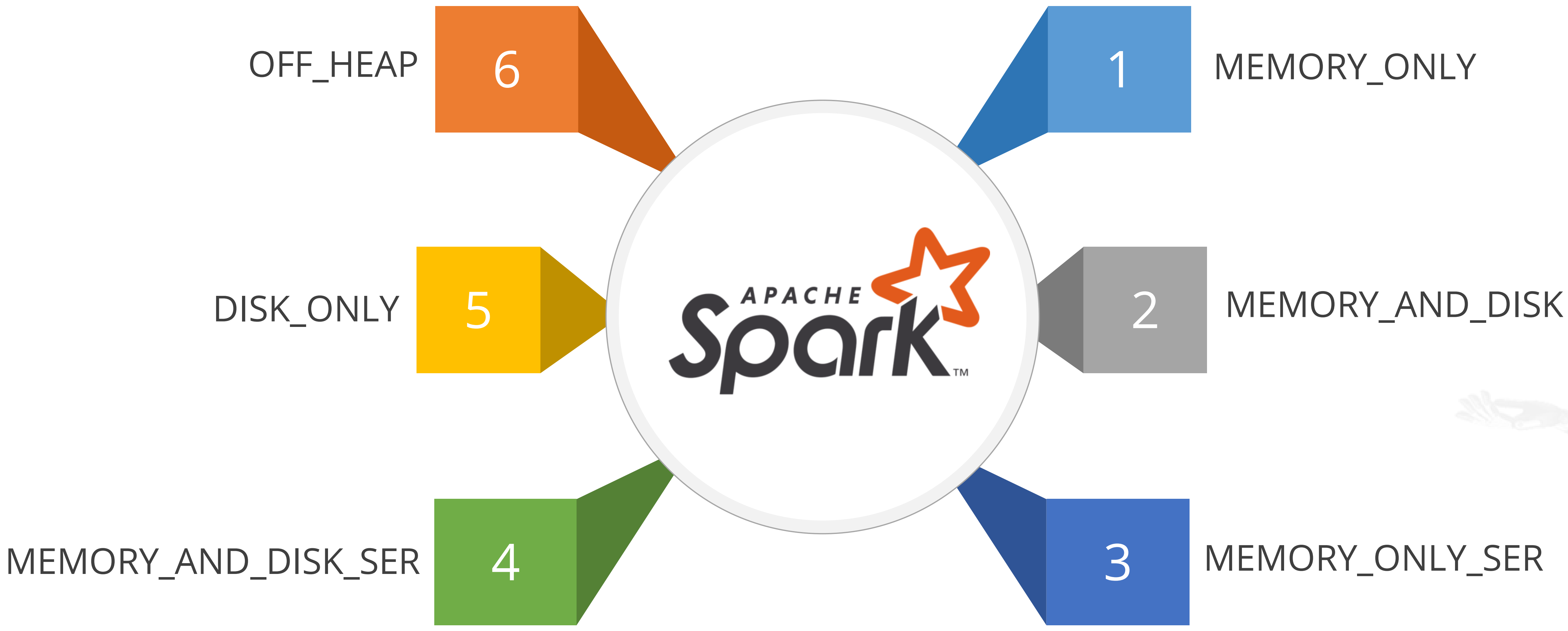
Automatic recomputation of lost RDD partitions

Storage of persisted RDDs on different storage levels

Methods of Caching and Persistence



Storage Levels



Marking an RDD for Persistence

training@localhost:~

File Edit View Search Terminal Tabs Help

training@localhost:~

```
> mydata = sc.textFile("simplilearn.txt")
> myrdd1 = mydata.map(lambda s: s.upper())
> myrdd1.persist()
> myrdd2 = myrdd1.filter(lambda
s:s.startswith('I'))
```

Step 01
Step 02
Step 03
Step 04

Simplilearn is an education provider.
It is based in San Francisco.
It has trained 450,000+ customers.
It offers 400+ professional courses.

File: simplilearn.txt

Step 01

RDD[1] (mydata)

Step 02

RDD[2] (myrdd1)

Step 03

RDD[3] (myrdd2)

Step 04

Changing Persistence Options

To change persistence option on an RDD:

Use rdd.Unpersist()

To change the RDD persistence to different storage level:

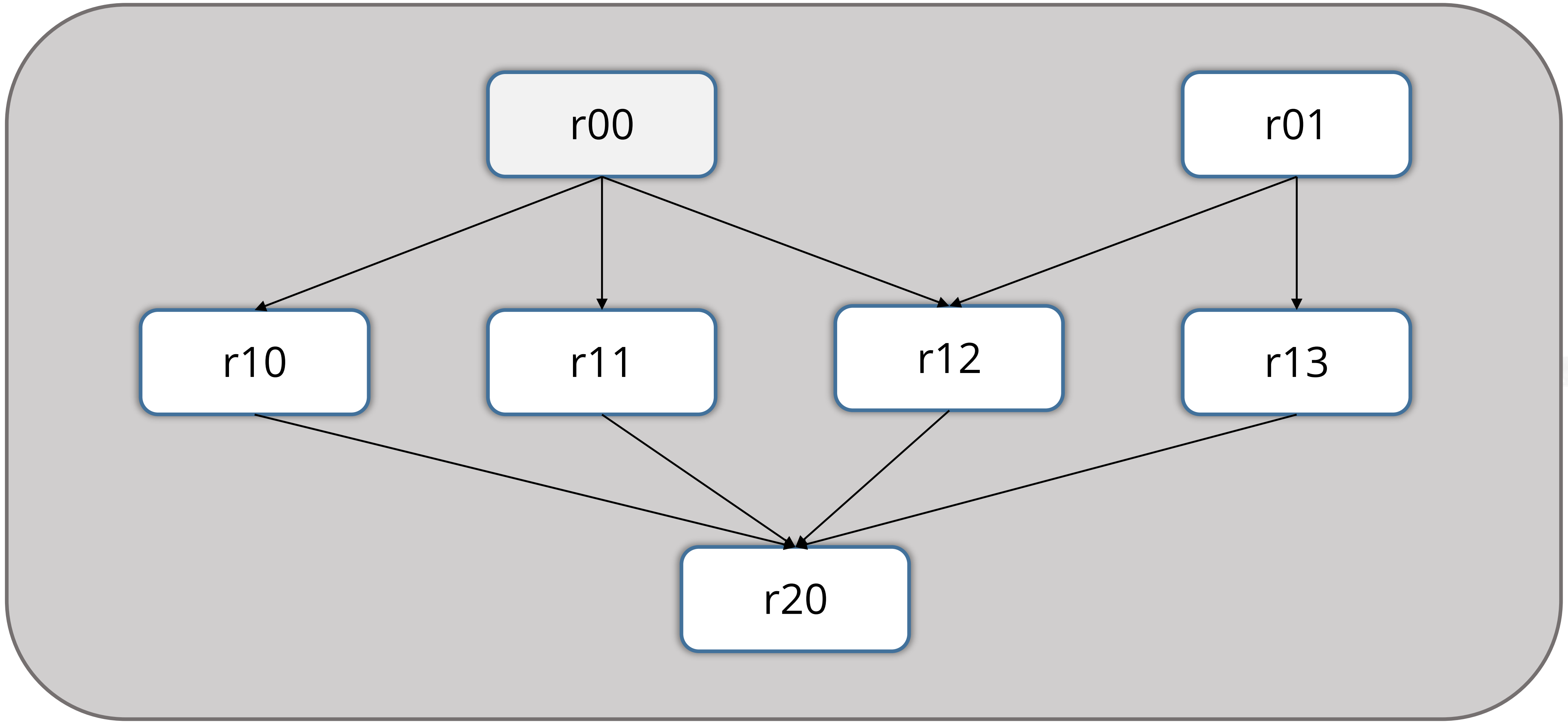
Unpersist the RDD and then mark it again for persistence with different storage level

DATA AND
ARTIFICIAL INTELLIGENCE

Lineage and DAG

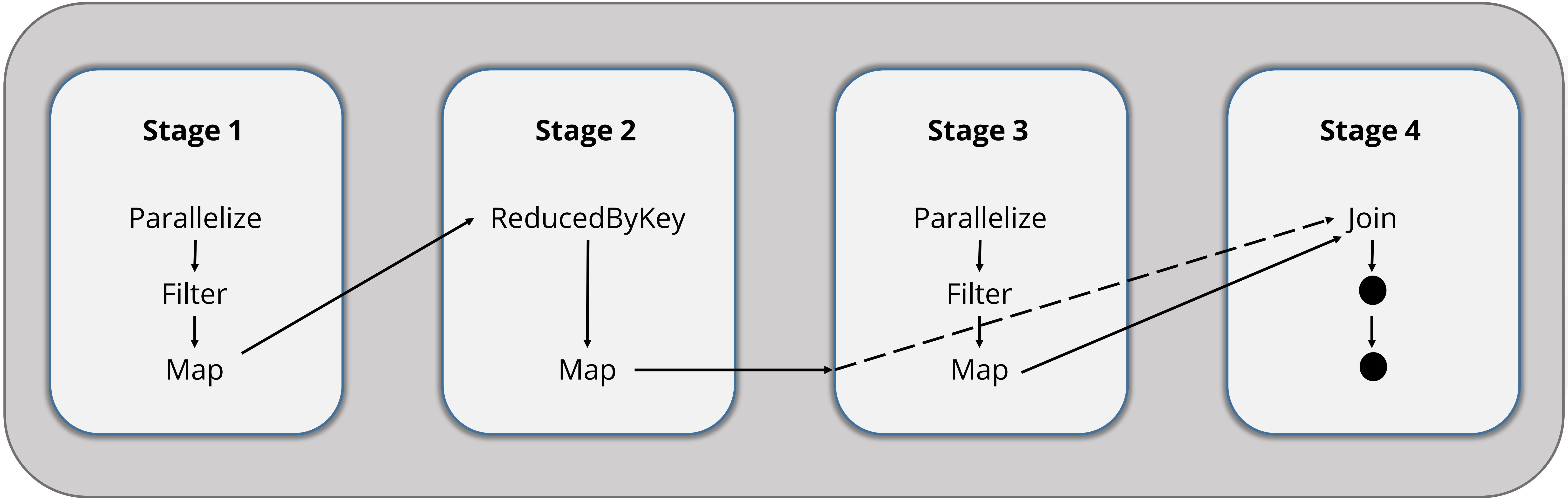
RDD Lineage

RDD Lineage is a graph that contains the existing RDD and the new RDD created from the existing one as a result of transformation.



DAG

Directed Acyclic Graph (DAG) is a graph where RDDs and the operations to be performed on RDDs are represented in the form of vertices and edges, respectively.

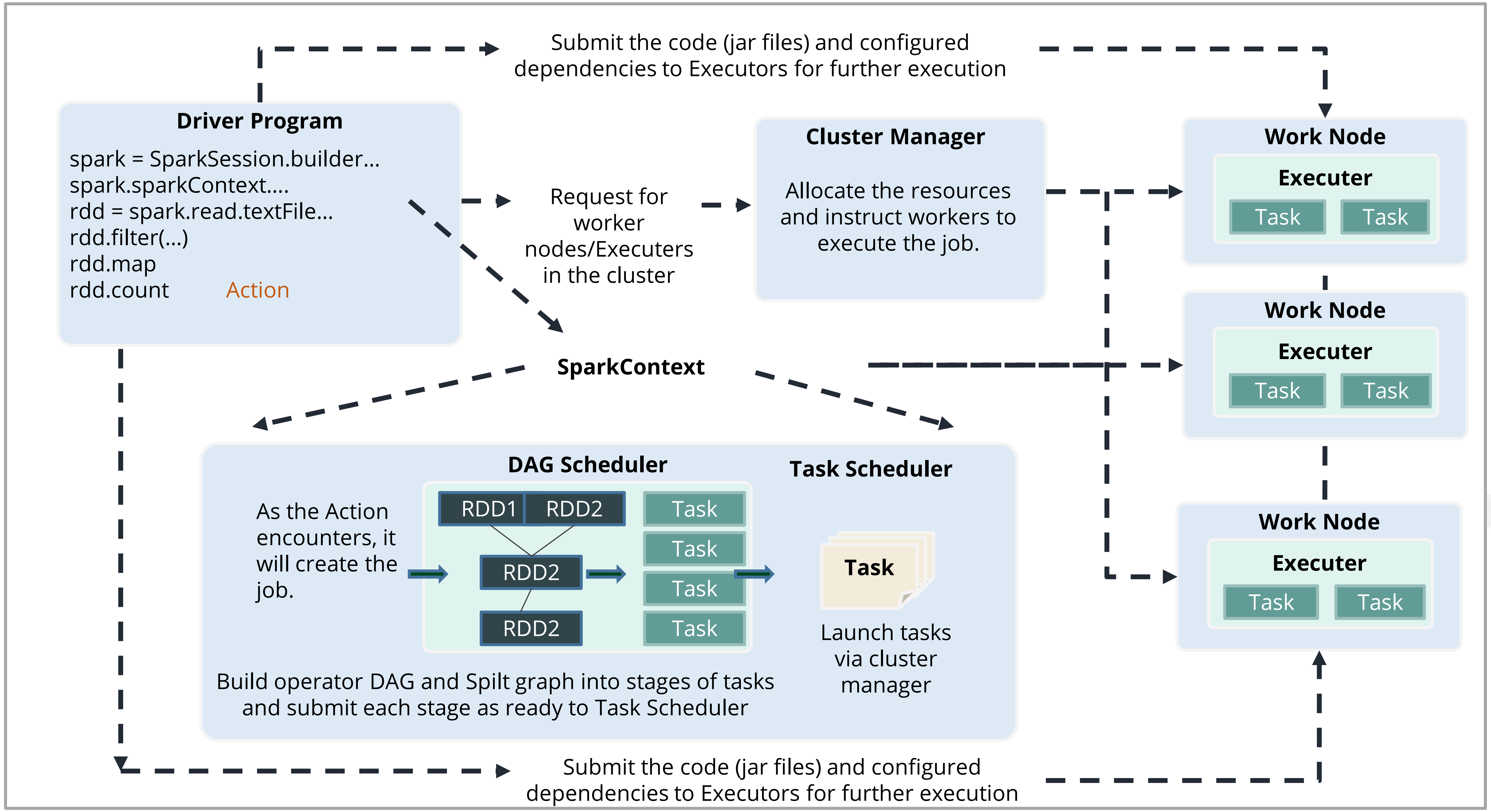


Need for DAG

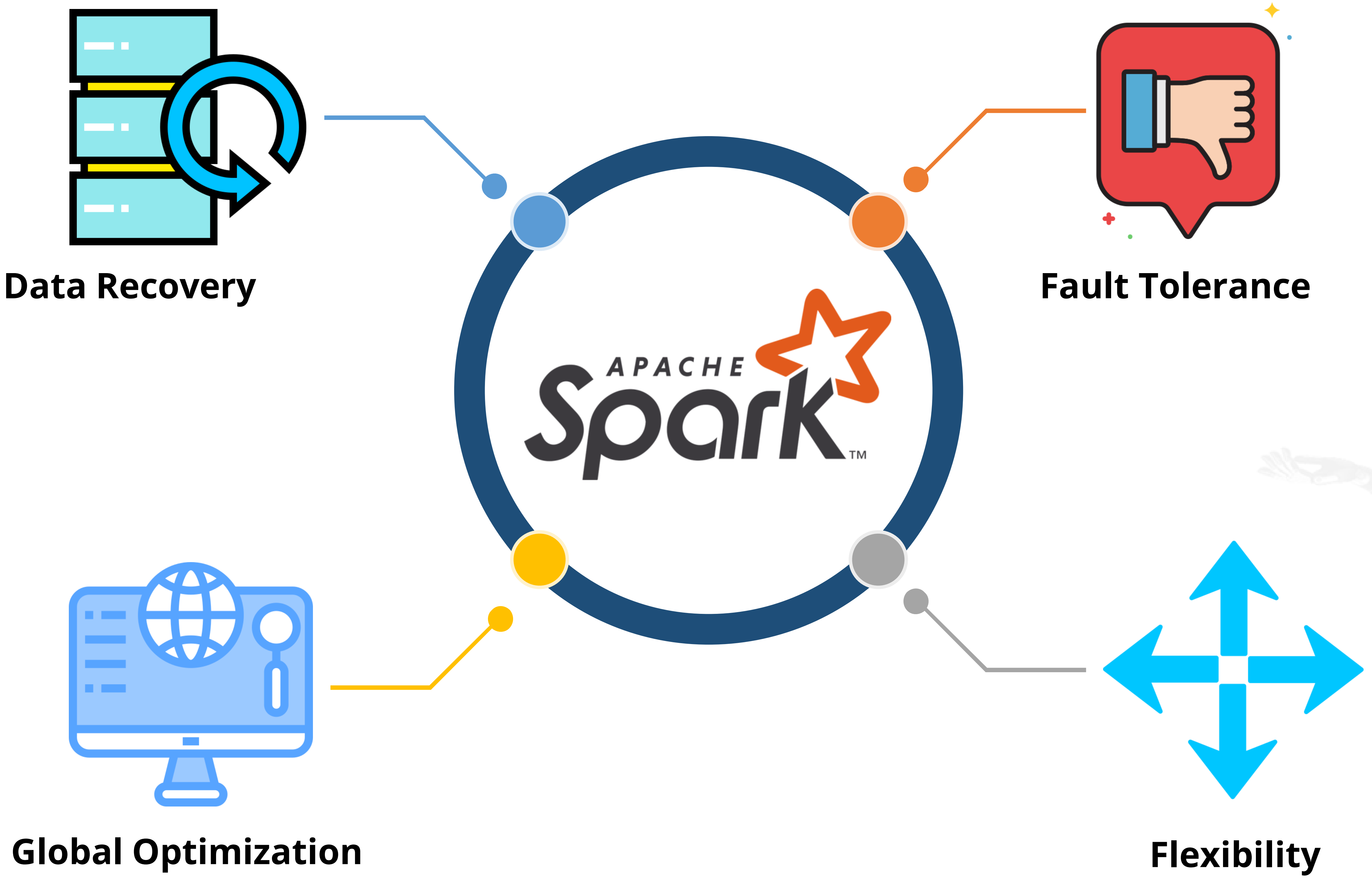
The need for DAG in Spark was created to overcome the limitations of DAG. The computation in MapReduce is done as:

1. The data is read from HDFS
2. Map and Reduce operations are applied to them
3. The result is written back to HDFS

Working of DAG



Advantages of DAG

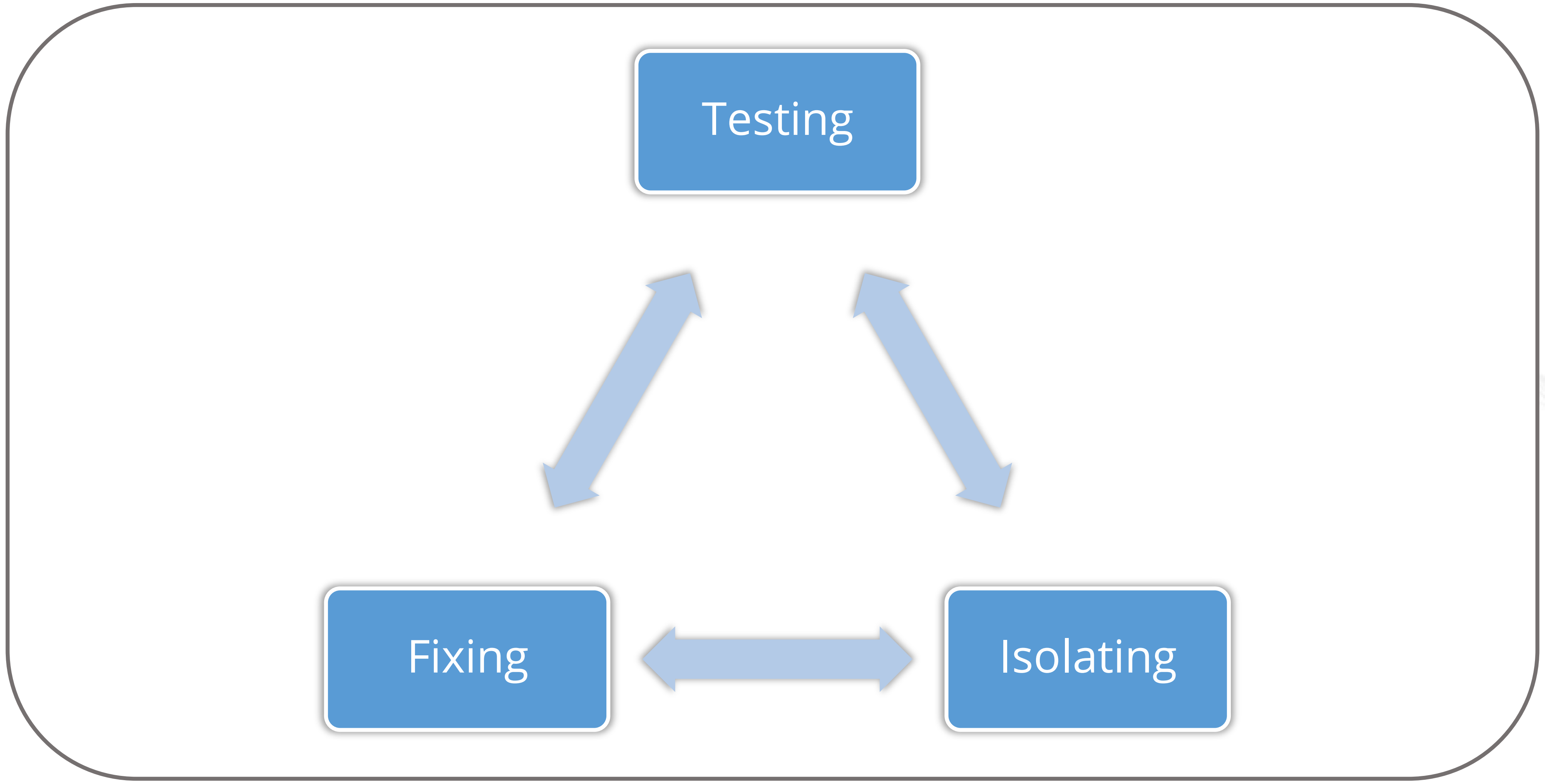


DATA AND
ARTIFICIAL INTELLIGENCE

Debugging in Spark

What Is Debugging?

Debugging is the process of identifying, isolating, and correcting a problem.



Attaching a Debugger to a Spark Application

The following are the steps for attaching a debugger to a Spark application:

Step 1	Upload the JAR file to the remote cluster and run it
Step 2	Define the SPARK_SUBMIT_OPTS environment variable and run it
Step 3	Connect to the debugger and configure it for use

DATA AND
ARTIFICIAL INTELLIGENCE

Partitioning in Spark

What Is Partitioning?

The data in an RDD is split into various segments called partitions.

01 Tuples in the same partition are guaranteed to be on the same machine.

02 Each machine contains one or more partitions.

03 The total number of partitions is configurable.

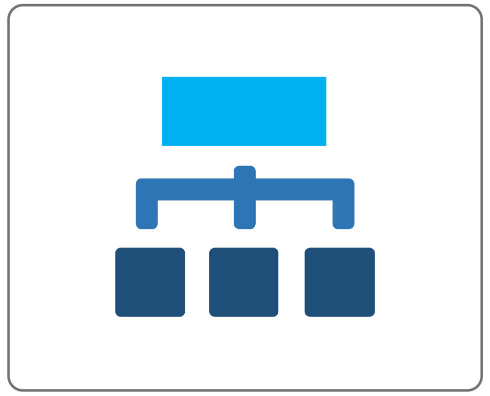
Types of Partitioning

Hash Partitioning

Range Partitioning

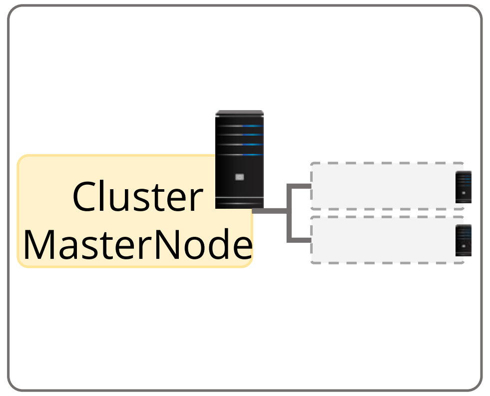
Custom Partitioning

Partitions from Single File

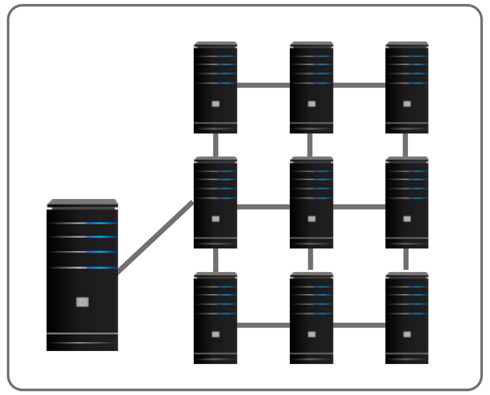


Partition can be done based on size

Partition can be done by specifying the minimum number of partitions as textFile(file, minPartitions)



While running on a cluster, the number of partitions by default is 2



More partitions = More parallelization

Partitions from Multiple Files

Partitions from multiple files can be done in the following ways:

sc.textFile("mydir/*")

sc.wholeTextFiles(“mydir”)

Operations on Partitions

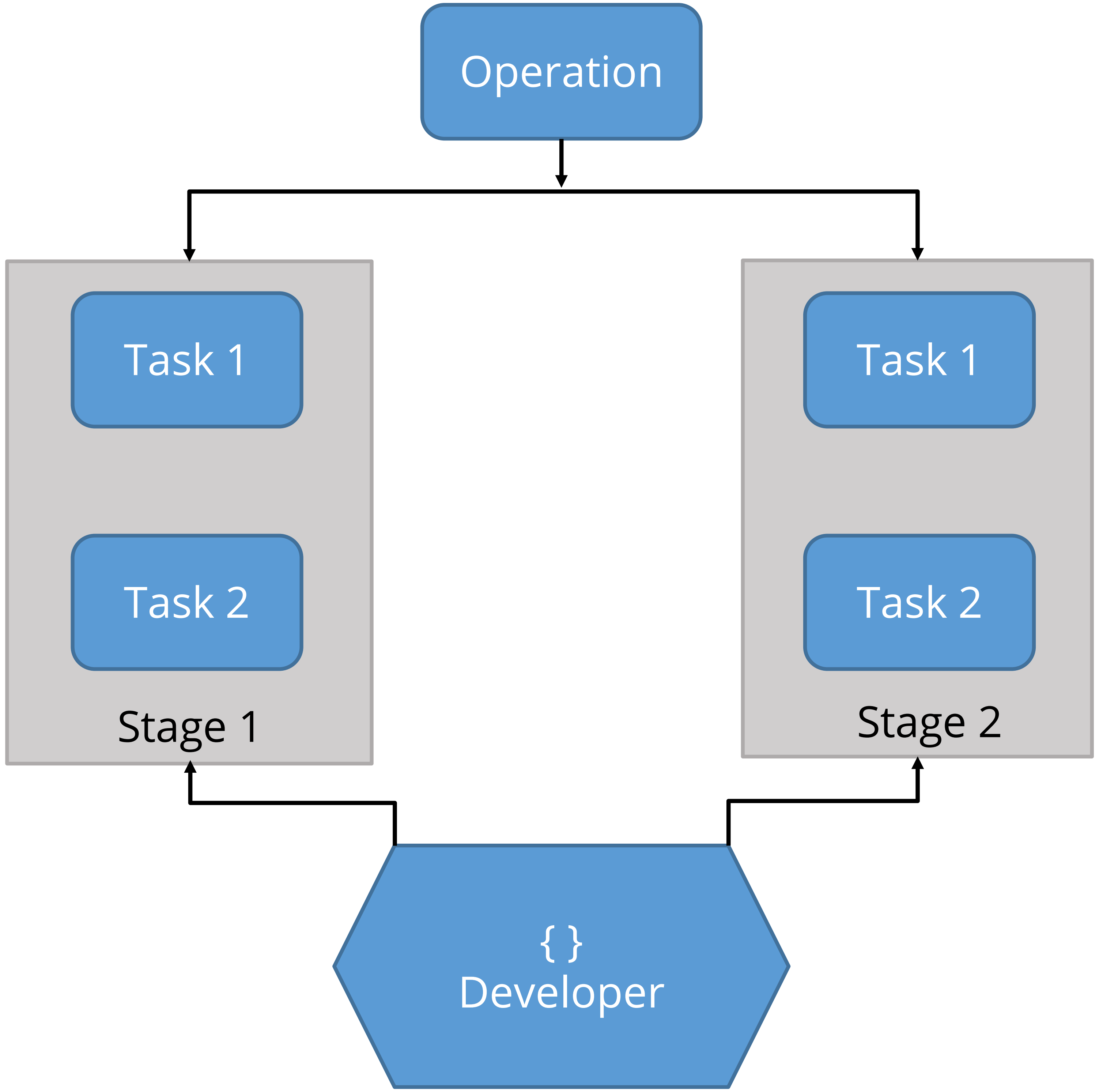
RDD operations work on each of the following partitions:

foreachPartition

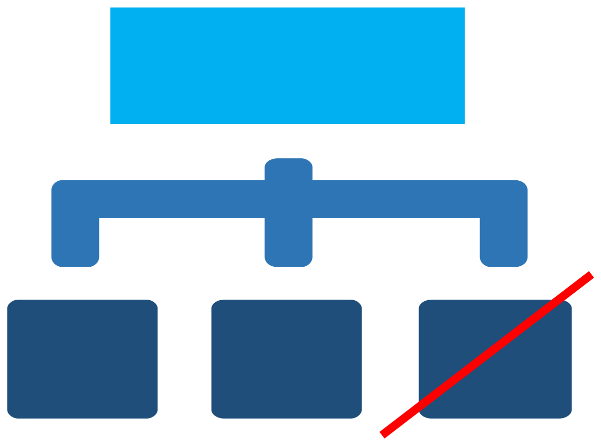
mapPartitions

mapPartitionsWithIndex

Operation Stages

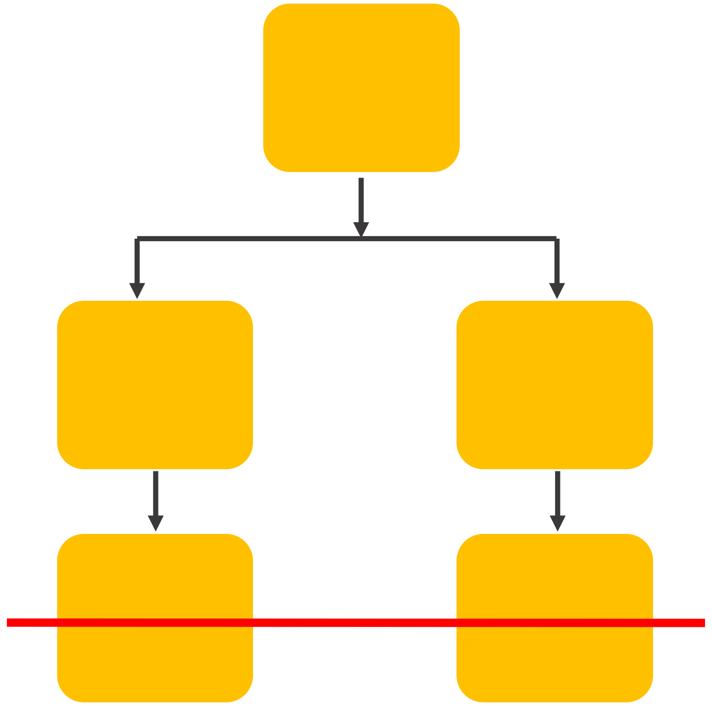


Parallel Operations on Stages



map
flatMap
filter
distinct

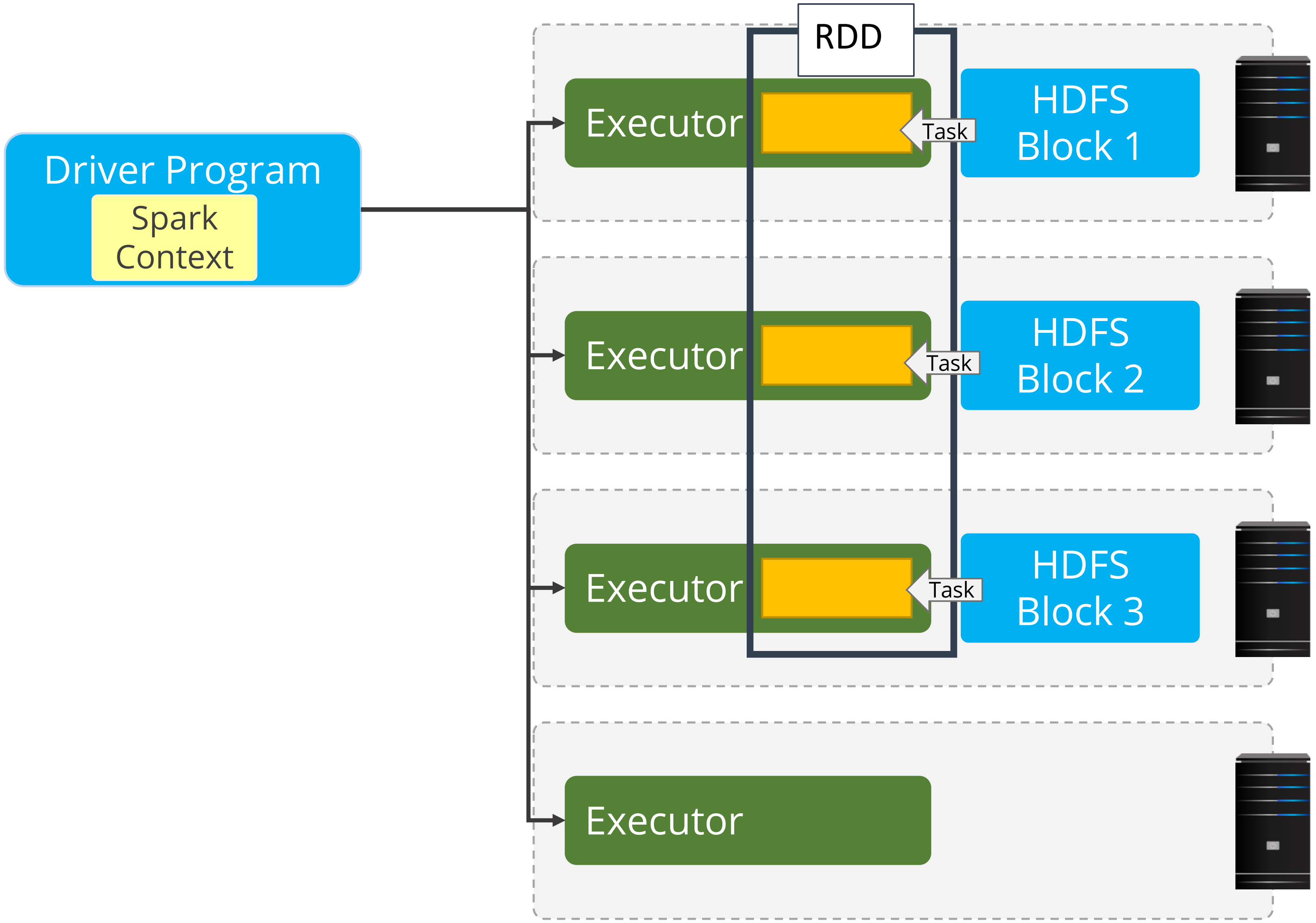
Some operations preserve partitioning



reduceByKey
sortByKey
join
groupByKey

Some operations perform repartition

Parallel Operations on Stages



HDFS:
mydata

DATA AND
ARTIFICIAL INTELLIGENCE

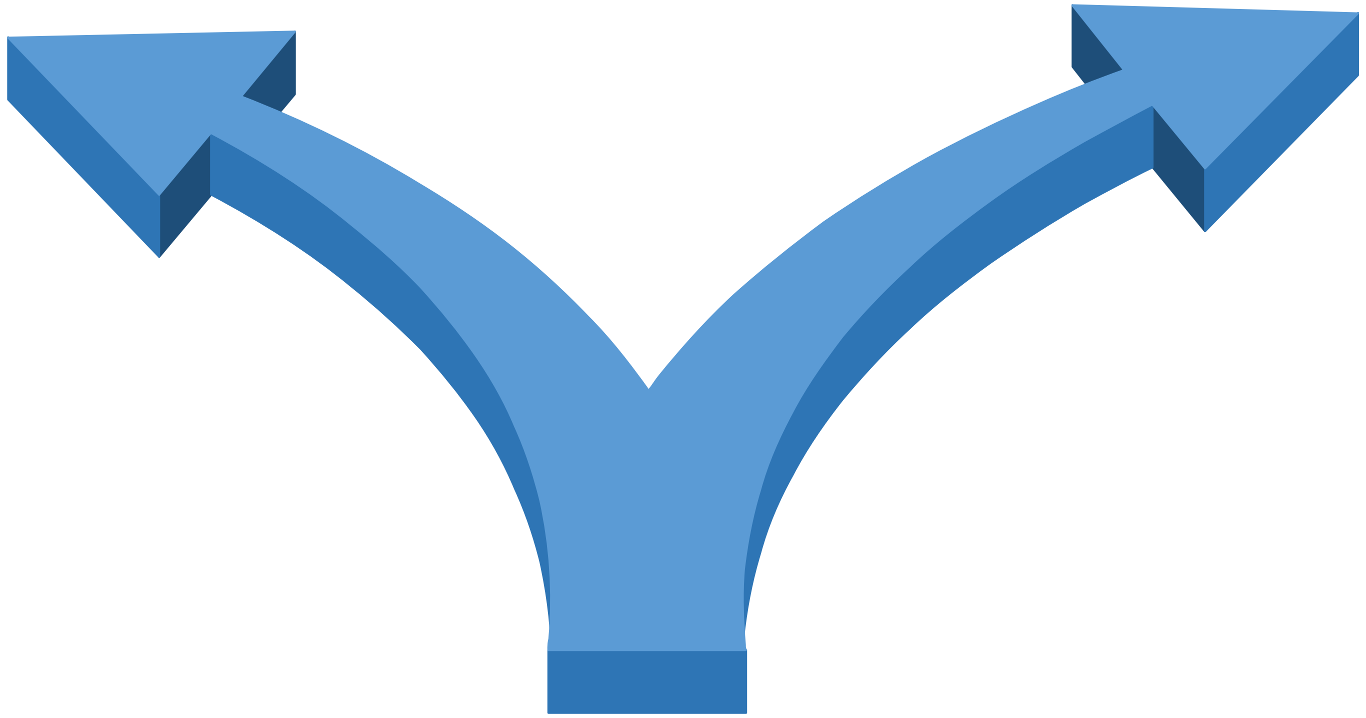
Scheduling in Spark

simplilearn

Scheduling in Spark

Scheduling is the process in which resources are allocated to different tasks by an operating system.

Scheduling across Applications

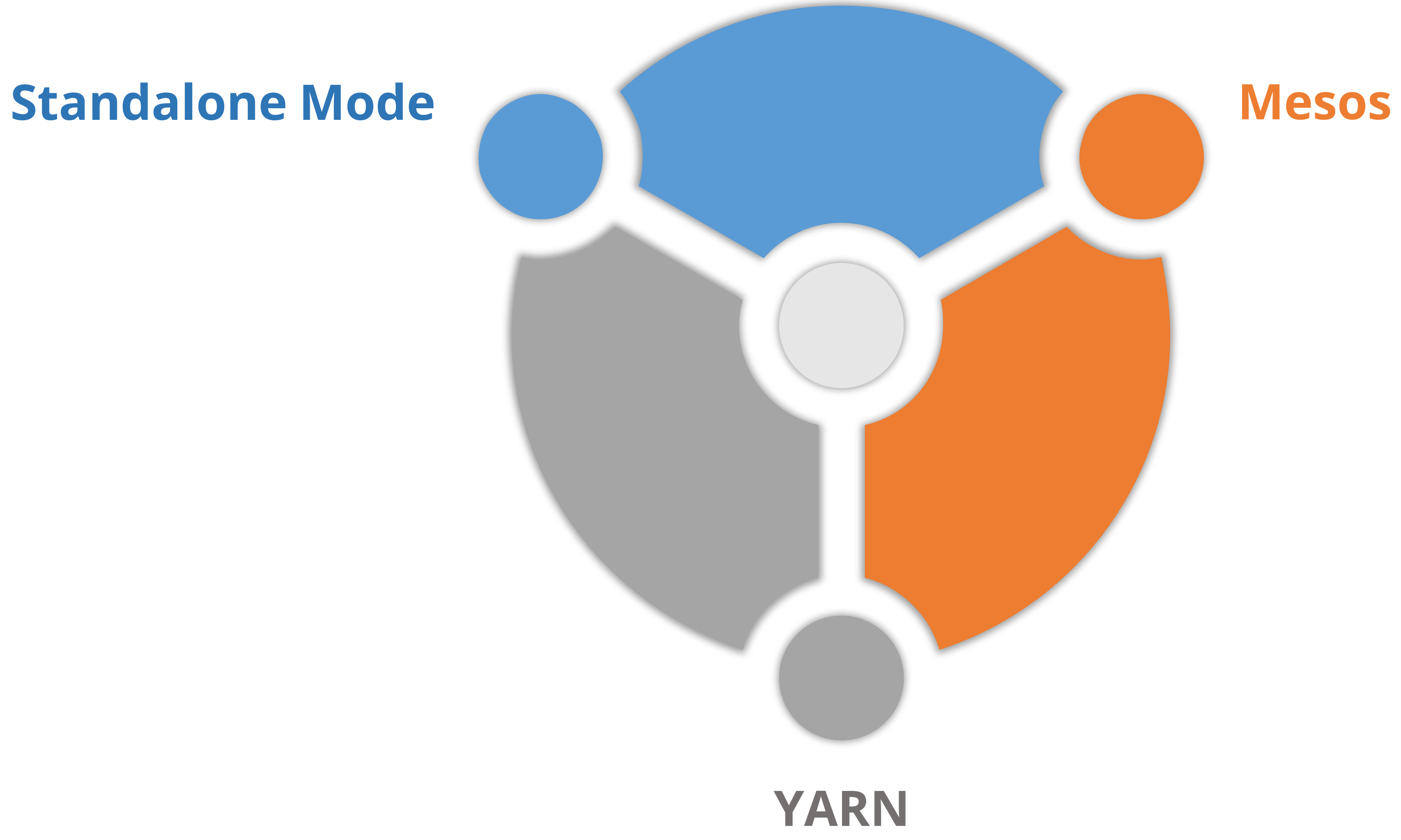


Scheduling within Applications

Scheduling across Applications

In Spark, each application has its own JVM that runs tasks and stores data.

Static partitioning is the best approach for allocating resources to each application in the following Spark modes:



Scheduling within Application

Multiple jobs can run simultaneously inside a Spark application, if they are submitted from separate threads.

Each job is divided into stages and resources are allocated in FIFO fashion.

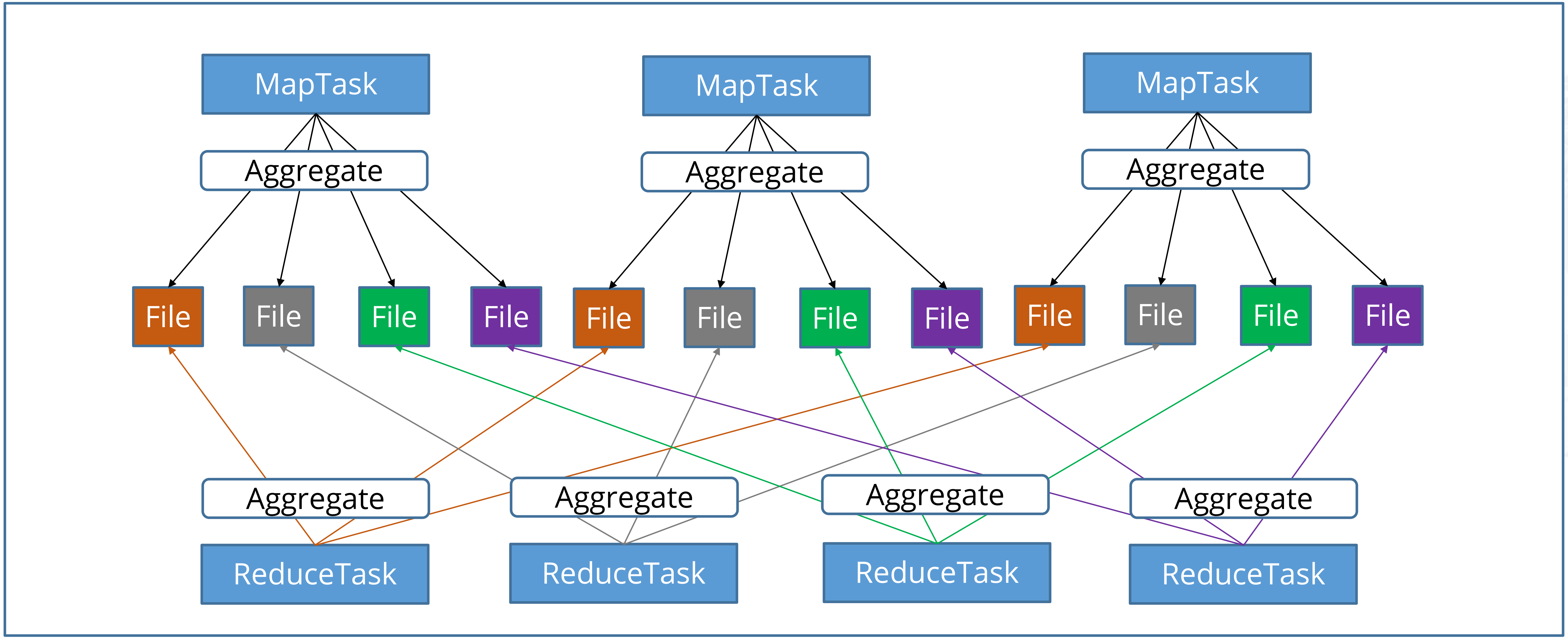
In Spark 8.0, it is possible to enable fair scheduler which uses round-robin fashion to allocate tasks between jobs.

```
val conf = new SparkConf().setMaster(...).setAppName(...)

conf.set("spark.scheduler.mode", "FAIR")

val sc = new SparkContext(conf)
```

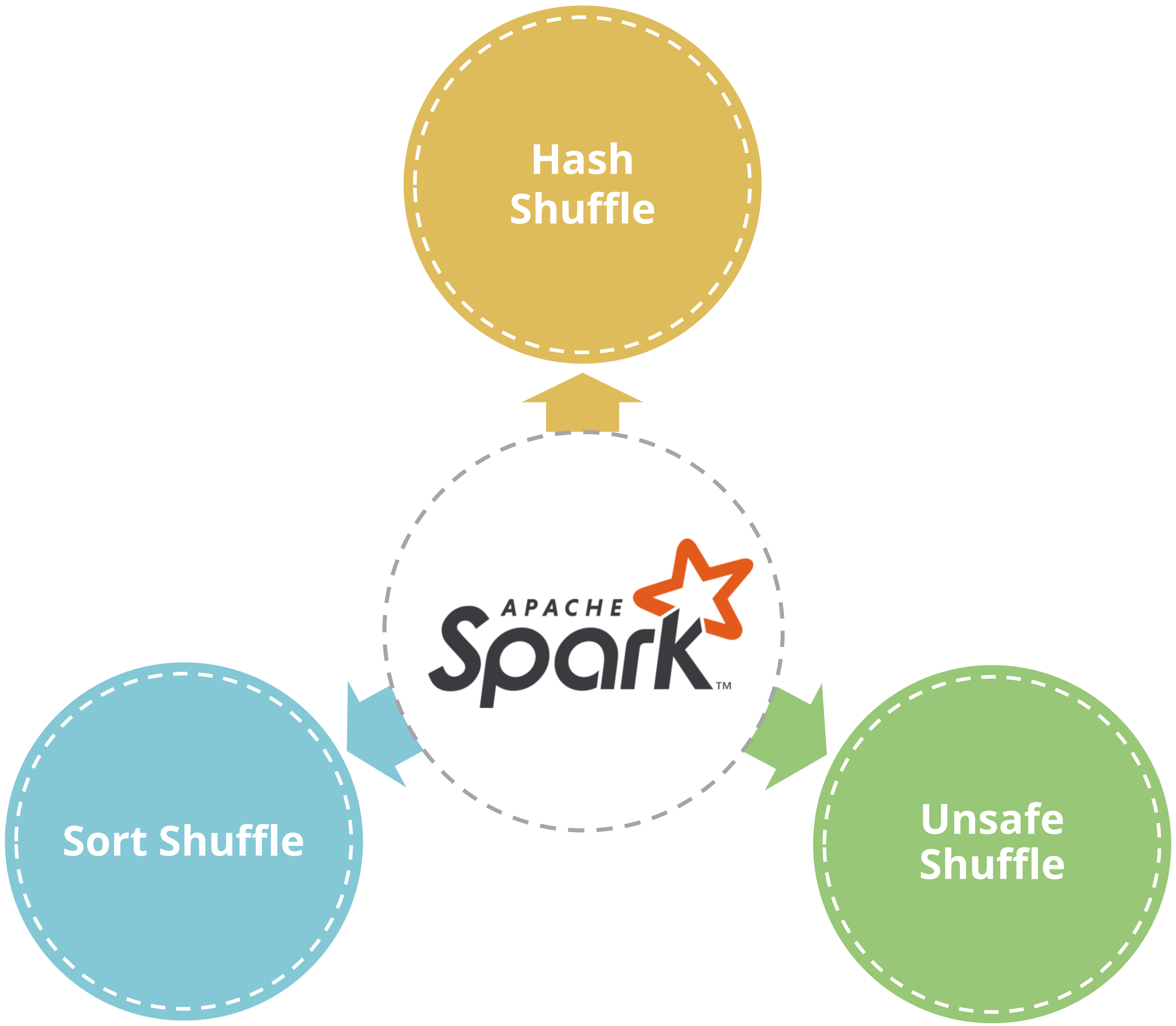
Shuffling in Spark

Shuffling in Spark

Shuffling is an operation which requires one node that will fetch data from other nodes to have data for computing the result.

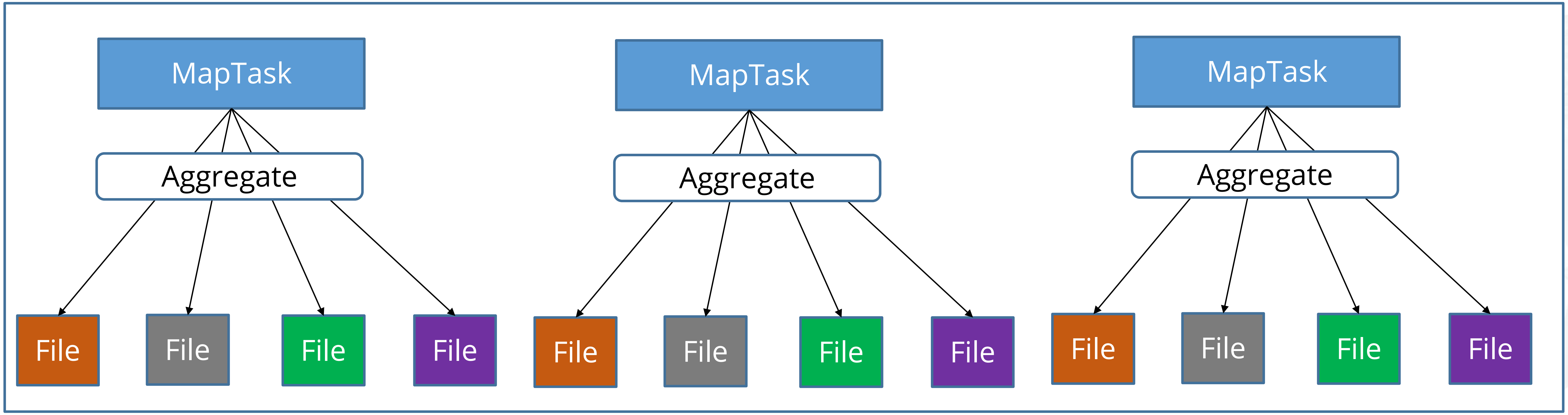


Shuffling in Spark

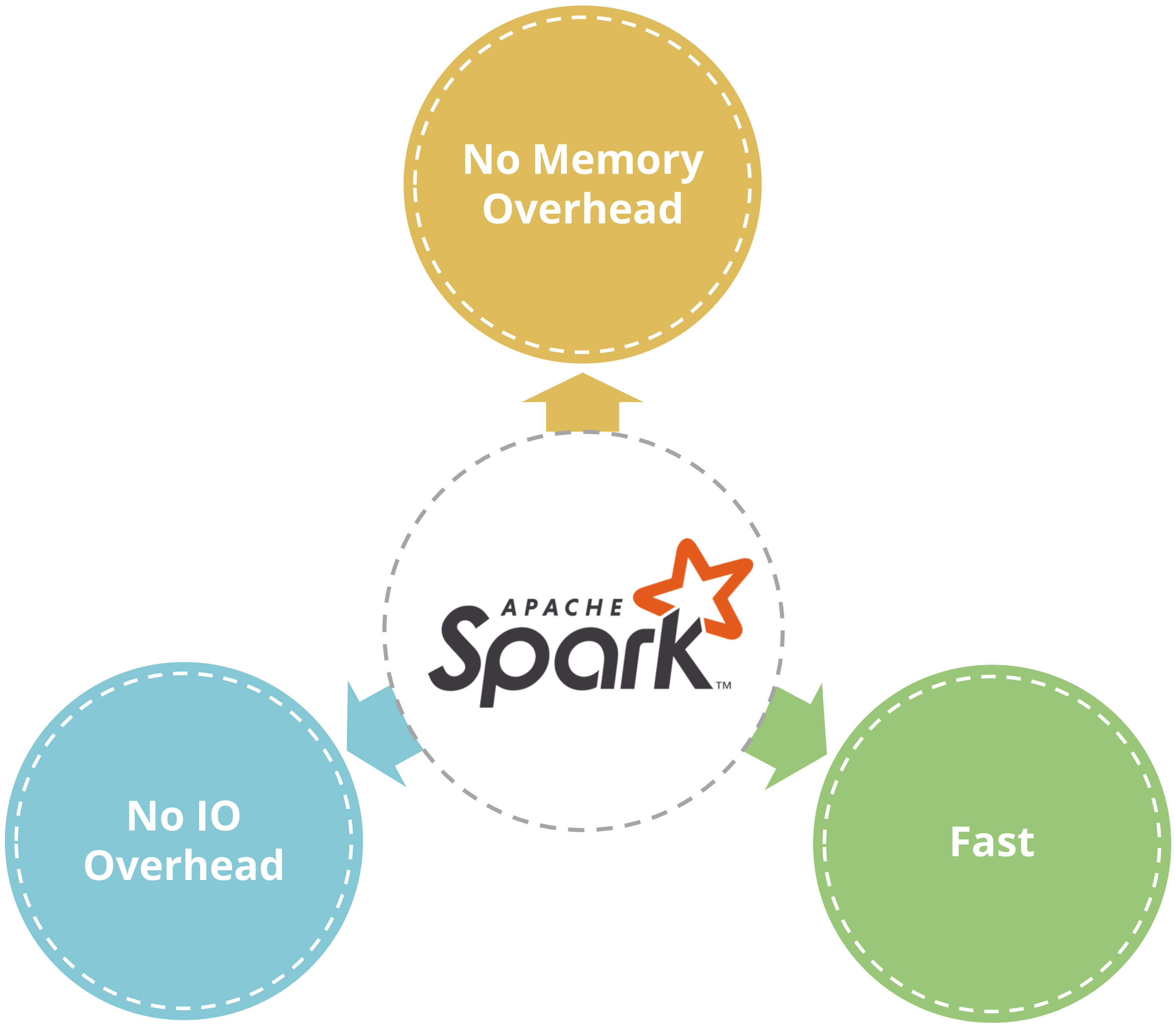


Hash Shuffle

In hash shuffle, each task will write the output into multiple files.

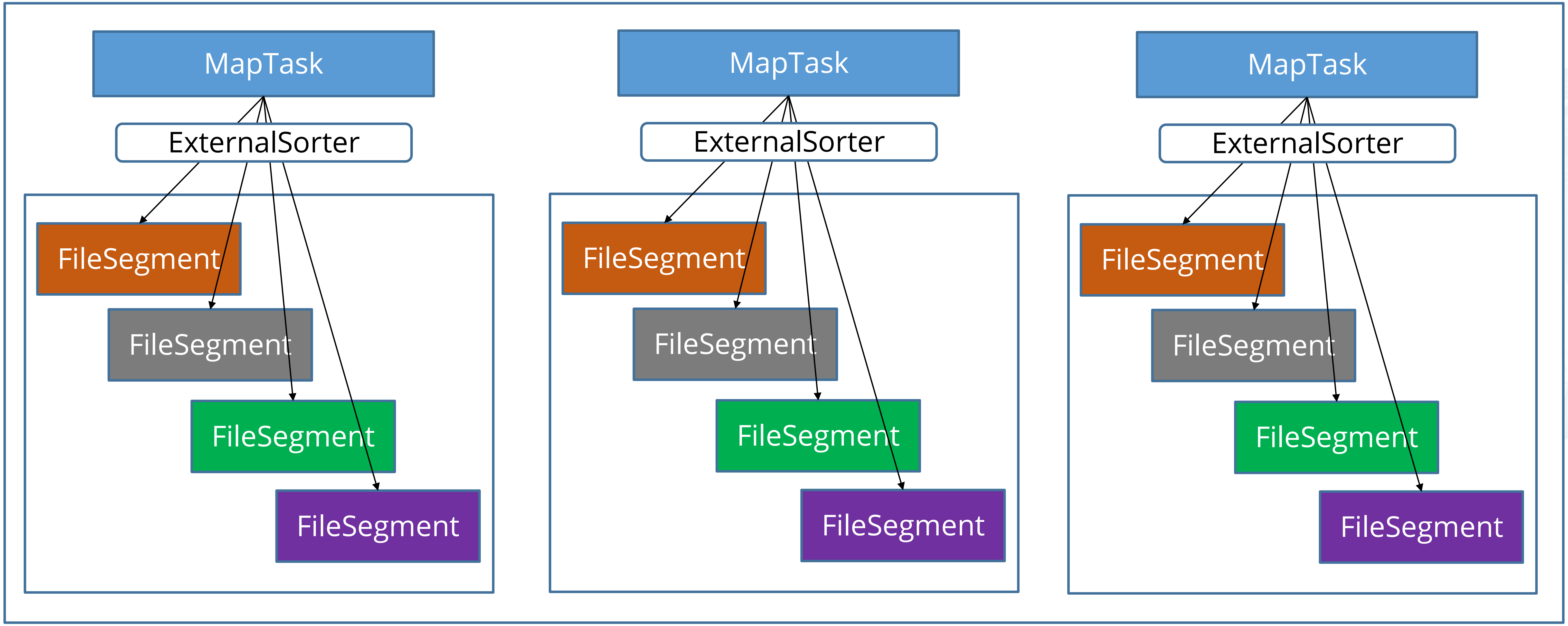


Advantages of Hash Shuffle



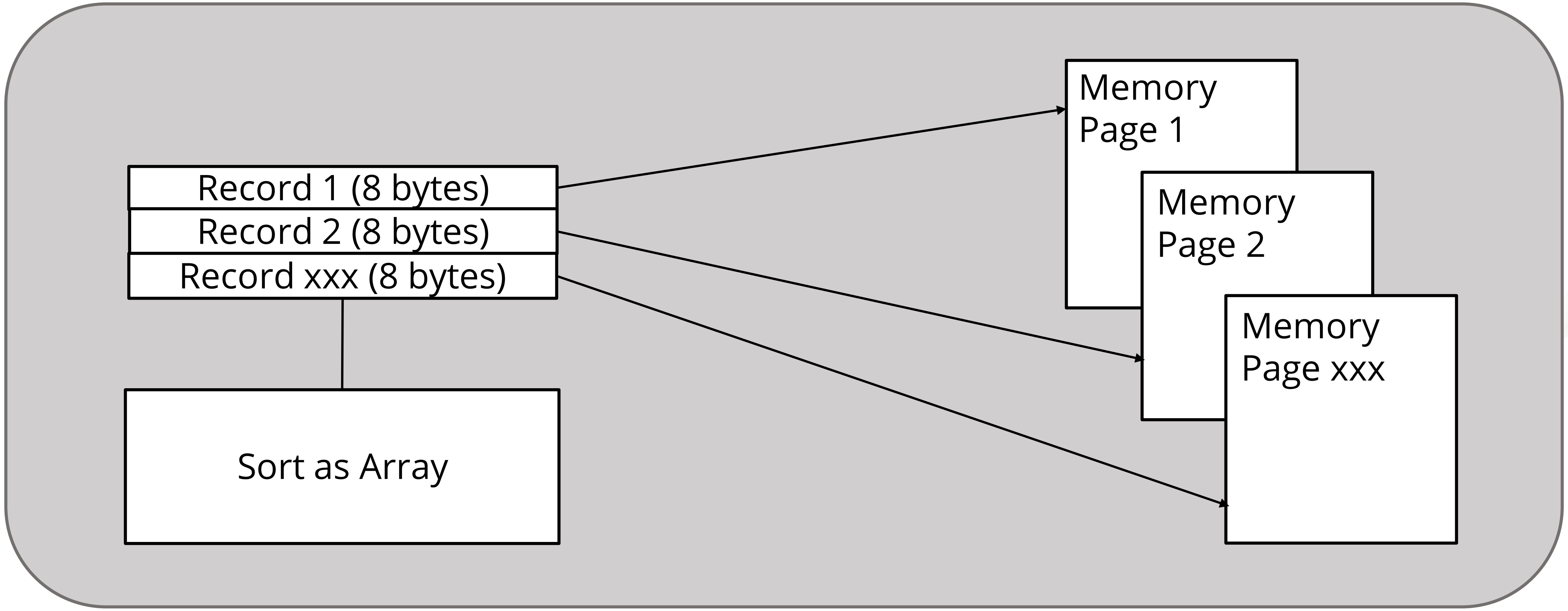
Sort Shuffle

In sort shuffle, each task spills only one shuffle containing segments and one index file.



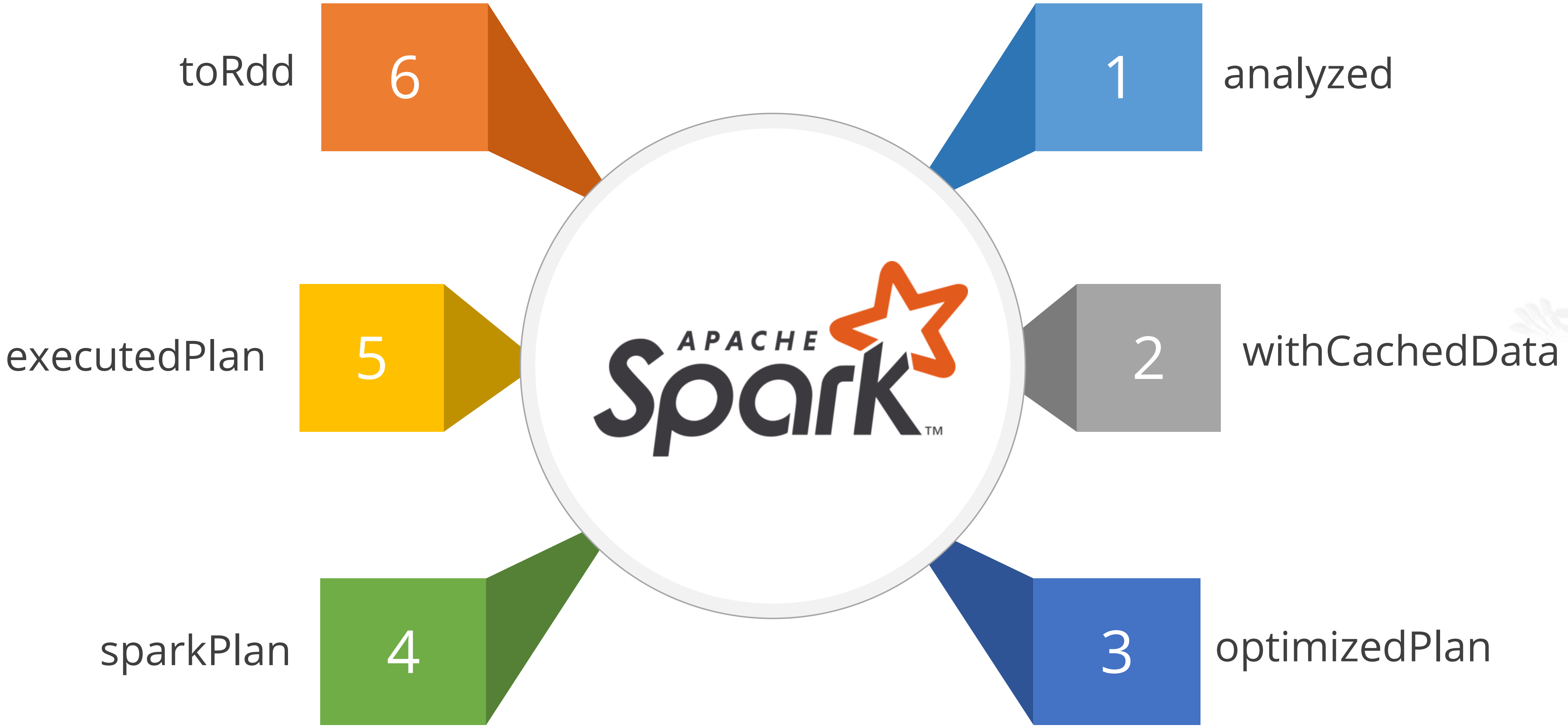
Unsafe Shuffle

In unsafe shuffle, the records are serialized once and then stored in memory pages.

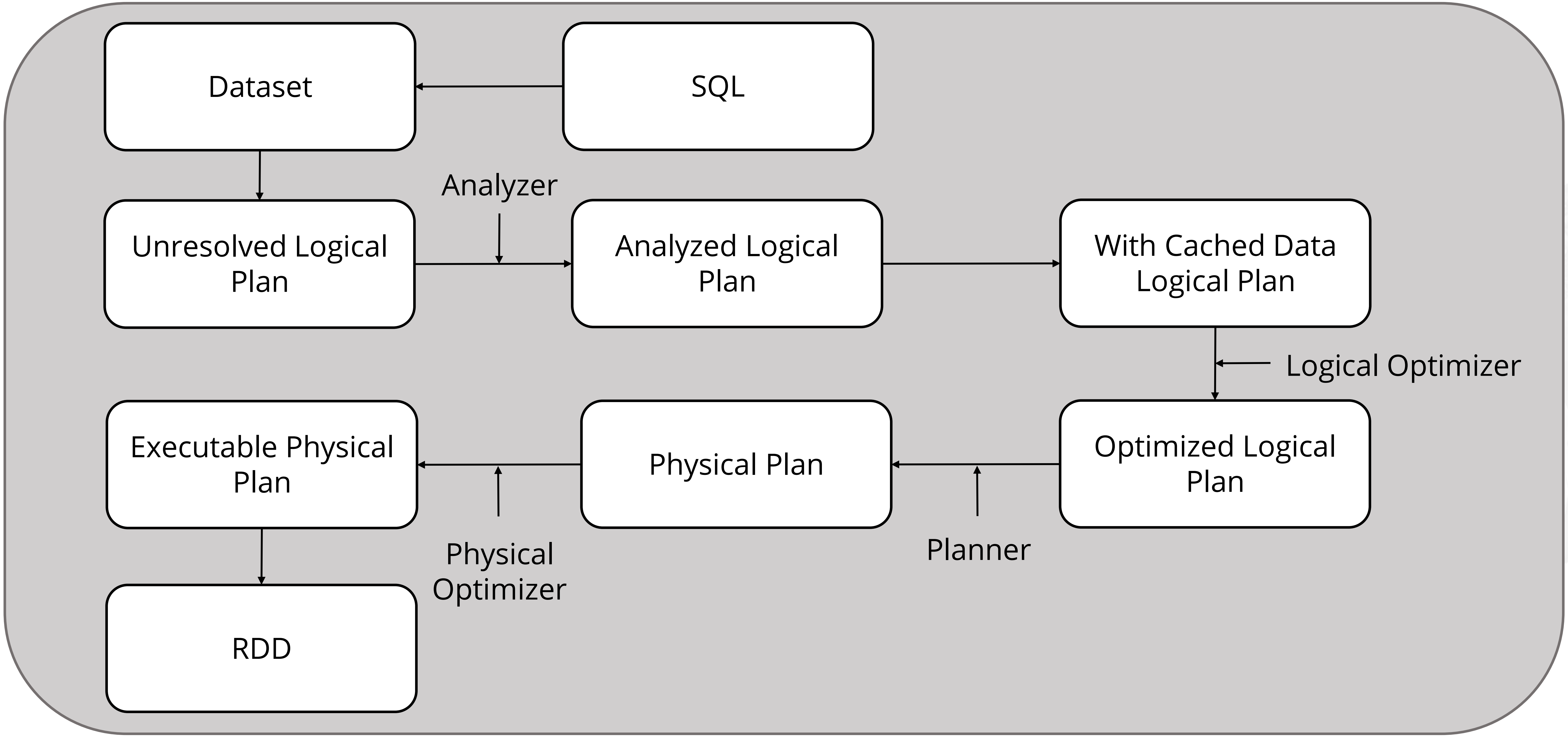


Query Execution in Spark

The following are the query execution phases in Spark:



Execution in Spark

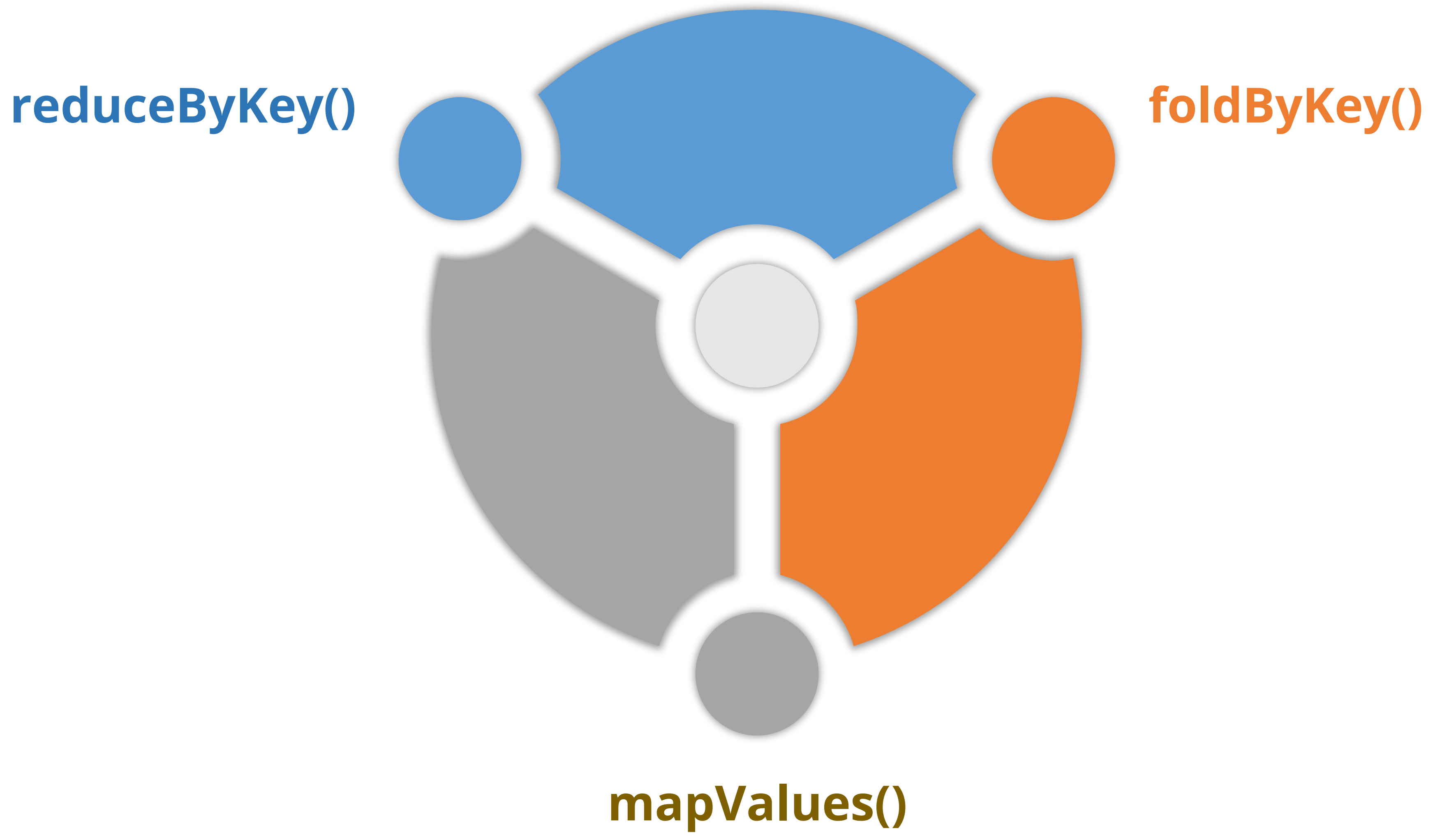


Aggregating Data with Pair RDD

Aggregation

To perform aggregations, the datasets must be described in the form of key-value pairs.

The following are the functions that can be used for aggregation:



Aggregation

Key	Value
Panda	0
Pink	3
Pirate	3
Panda	1
Pink	4

mapValues()

Key	Value
Panda	(0,1)
Pink	(3,1)
Pirate	(3,1)
Panda	(1,1)
Pink	(4,1)

reduceByKey()

Key	Value
Panda	(1,2)
Pink	(7,2)
Pirate	(3,1)

Spark Application with Data Written Back to HDFS and Spark UI

Duration: 10 mins

Problem Statement: In this demonstration, you will write the data to HDFS and Spark UI.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Changing Spark Application Parameters

Duration: 10 mins

Problem Statement: In this demonstration, you will change Spark application parameters.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Handling Different File Formats

Duration: 10 mins

Problem Statement: In this demonstration, you will handle different file formats.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Spark RDD with Real-World Application

Duration: 10 mins

Problem Statement: In this demonstration, you will understand Spark RDD with a real-world application.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Optimizing Spark Jobs

Duration: 10 mins

Problem Statement: In this demonstration, you will optimize Spark jobs.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Key Takeaways

You are now able to:

- Define Spark RDD and list its limitations
- Describe and demonstrate RDD operations in Spark
- Demonstrate the creation of Spark RDD
- Aggregate data with pair RDD

Knowledge Check

Knowledge Check 1

Which feature of Spark RDD applies to all elements in datasets through maps, filter, or group by operation?

a. Lazy Evaluation

b. Cross-Grained Operation

c. Fault-Tolerant

d. Immutable

simplilearn

Knowledge Check 1

Which feature of Spark RDD applies to all elements in datasets through maps, filter, or group by operation?

a. Lazy Evaluation

b. Cross-Grained Operation

c. Fault-Tolerant

d. Immutable

The correct answer is **b**

Cross-Grained Operation feature of Spark RDD applies to all elements in datasets through maps, filter, or group by operation.

Knowledge Check 2

Which of the following transformations is not self-sufficient?

a. Narrow Transformation

b. Wide Transformation

c. Both a and b

d. None of the above

Knowledge Check 2

Which of the following transformations is not self-sufficient?

a. Narrow Transformation

b. Wide Transformation

c. Both a and b

d. None of the above

The correct answer is **b**

Wide transformation is not self-sufficient.

Knowledge Check 3

Which of the following query execution phases makes sure that the logical plan is analyzed and uses the supported operations only?

a. toRdd

b. executedPlan

c. withCachedData

d. optimizedPlan

Knowledge Check 3

Which of the following query execution phases makes sure that the logical plan is analyzed and uses the supported operations only?

a. toRdd

b. executedPlan

c. withCachedData

d. optimizedPlan

The correct answer is **c**

withCachedData makes sure that the logical plan is analyzed and uses the supported operations only.

Lesson-End Project

Problem Statement: The New York School authority collects data from all schools that provide bus facilities to students. This data helps to understand if buses are reaching on time or not. This also helps to understand if there is a specific route where buses are taking more time so that it can be improved.

You are given a dataset of buses which got broke down or are running late. You have been given the below data set:

Filename: bus-breakdown-and-delays.csv

1. School_Year
   a. Indicates the year the record refers to
2. Run_Type
   a. Designates whether a breakdown or delay occurred on a specific category of bus services
3. Bus_No
4. Route_Number
5. Reason
   a. Reason for delay as entered by the staff employed by the reporting bus vendor
6. Occurred_On
7. Number_Of_Students_On_The_Bus

Lesson-End Project

You are hired as a big data consultant to provide important insights. You must write a Spark job using the above data and need to provide the following:

1. Most common reasons for either a delay or breaking down of bus
2. Top 5 route numbers where the bus was either delayed or broke down
3. The total number of incidents,year-wise, when the students were
a. In the bus
b. Not in the bus
4. The year in which accidents were less