upGrad

Big Data Engineering
Apache Spark

Course: Platforms for Big Data

Live Session on: Apache Spark

Speaker: Abhinav Rawat

upGrad

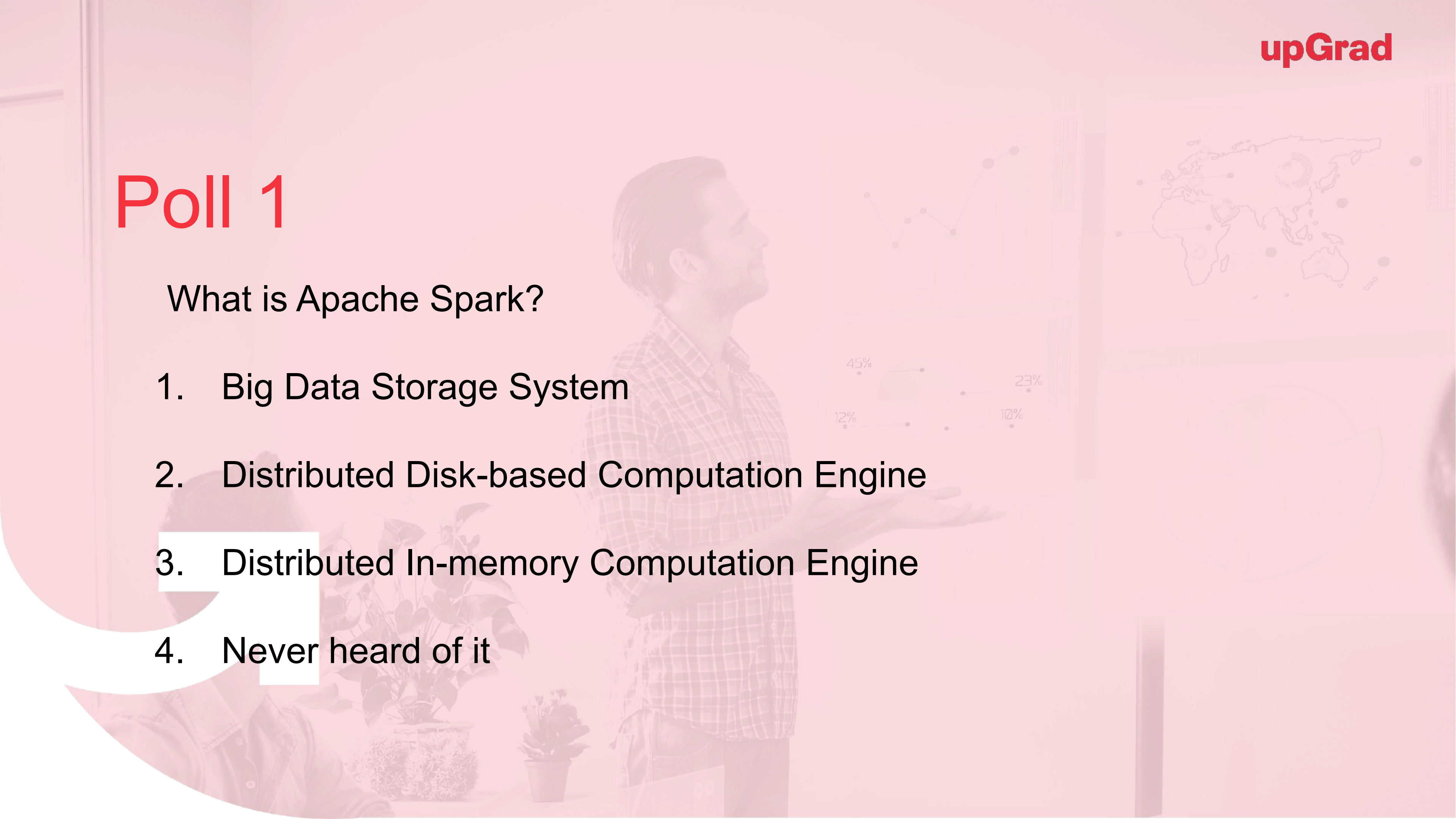
Today’s Agenda

- Introduction to Spark
- Map-Reduce vs. Spark
- Spark Ecosystem
- Spark Architecture
- Spark RDDs
- Spark SQL: Dataframe and Datasets
- Transformation and Actions
- Lazy Evaluation
- Advanced Concepts
- Spark in the Industry
- Limitation of Spark
- Coding Exercise
- Q & A

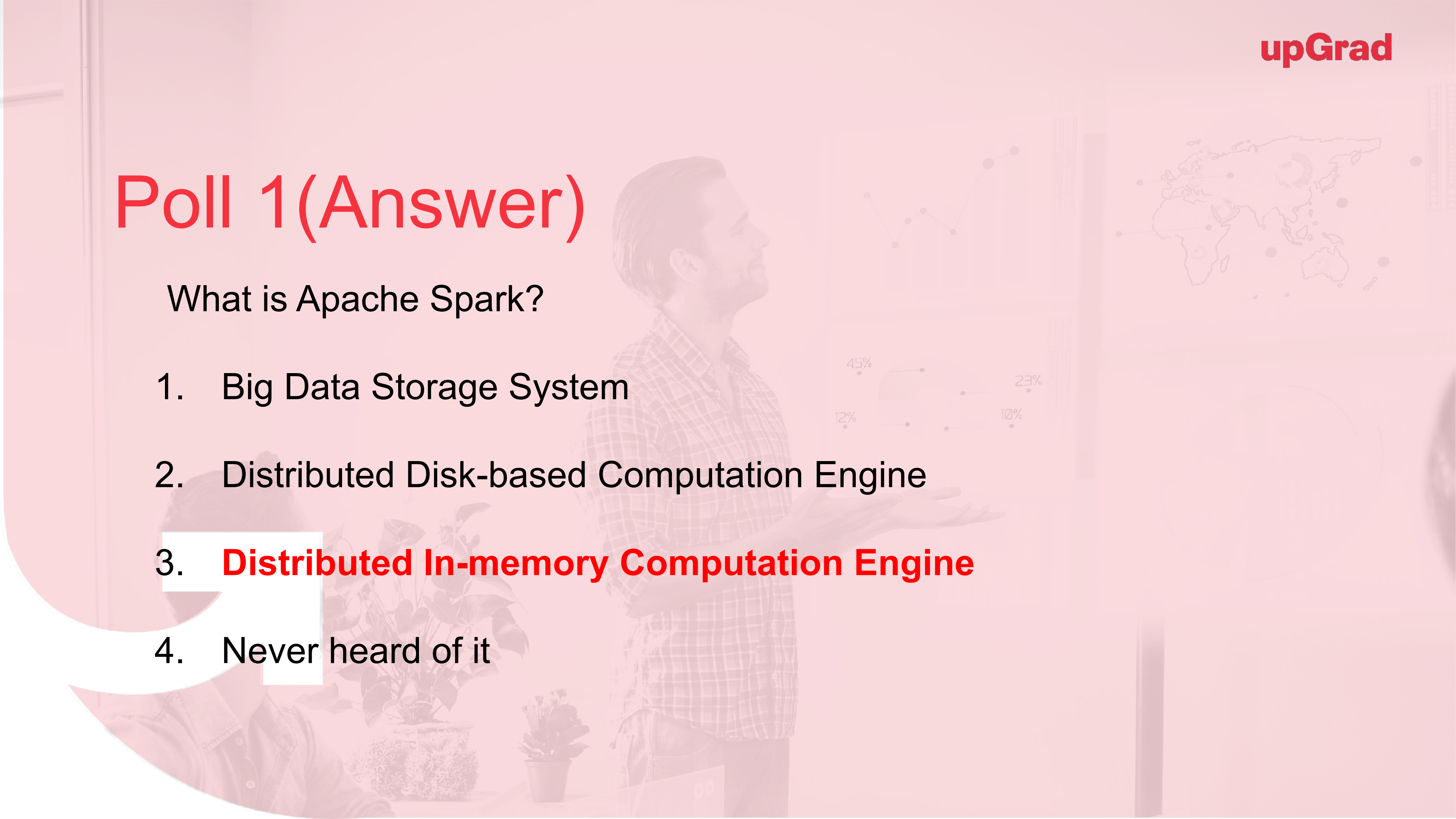
Poll 1

What is Apache Spark?

1. Big Data Storage System
2. Distributed Disk-based Computation Engine
3. Distributed In-memory Computation Engine
4. Never heard of it



upGrad



Poll 1(Answer)

What is Apache Spark?

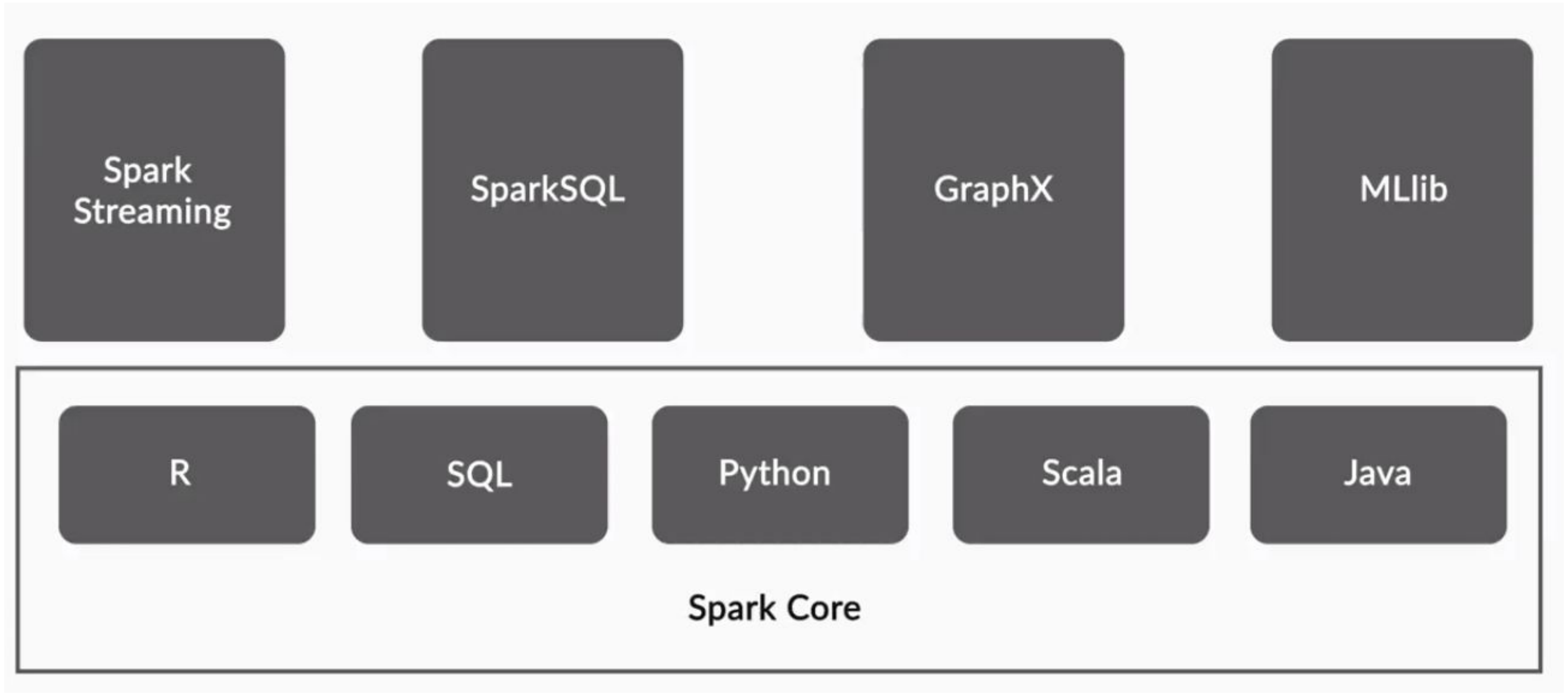
1. Big Data Storage System
2. Distributed Disk-based Computation Engine
3. **Distributed In-memory Computation Engine**
4. Never heard of it

Apache Spark: An Overview

- Distributed in memory data processing engine
- The three major reasons why Spark has gained popularity today are -
  - **Speed**: Due to in-memory processing, Spark processes data with lightning fast speed.
  - **Generality**: Spark can be used for a variety of tasks such as batch, interactive, streaming, etc. It also supports APIs in various languages such as Python, Java, Scala, etc.
  - **Runs Everywhere**: You can run spark application everywhere like local machine, Hadoop, Apache Mesos, Spark StandAlone

Apache Spark: Ecosystem

upGrad



Apache Spark: Ecosystem

upGrad

Spark Core
- Spark Core is the central processing engine of the Spark application.
- Spark Core makes use of a data structure known as RDDs

Spark SQL
- Spark SQL allows users to run SQL queries on top of Spark.
- Spark SQL also introduced DataFrame and datasets that could impose a schema on RDDs, which are schemaless.

Spark Streaming
- The Spark Streaming module processes streaming data in real time.
- Some of the popular streaming data sources are web server logs, data from social networking websites, such as Twitter feed, etc.

Spark MLlib
- MLlib is a machine learning library that is used to run machine learning algorithms on big data sets.
- It provides APIs for common machine learning algorithms such as clustering, classification, and generic gradient descent.

Spark GraphX
- The GraphX API allows a user to view data as a graph and combine graphs with RDDs.
- It provides an API for common graph algorithms such as Connect components and PageRank.

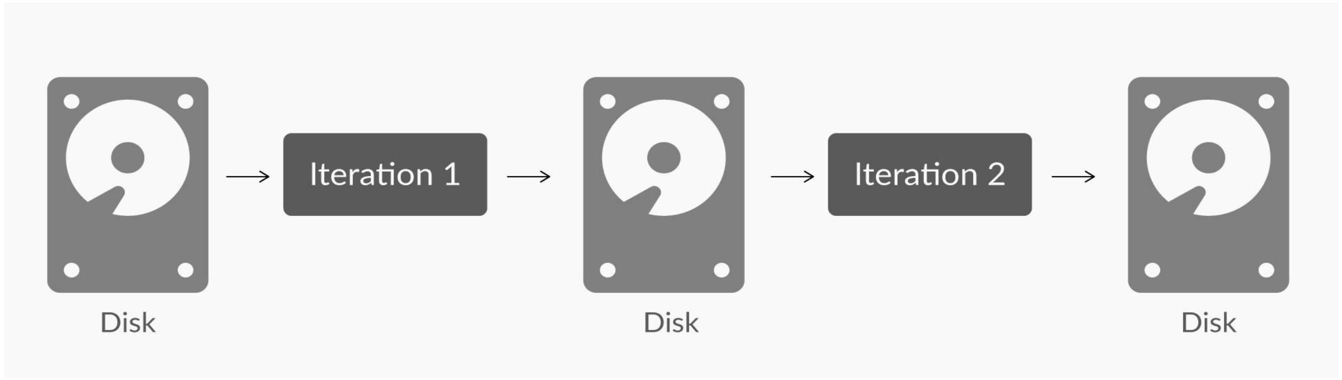
Map-Reduce vs. Spark

- Map-Redcue is purely a disk based processing system
- The intermediate output between of Map task also gets stored in the disk
- Excessive disk I/O operations
- Map-Reduce vs. Spark
    - Iterative Processing
    - Interactive Processing

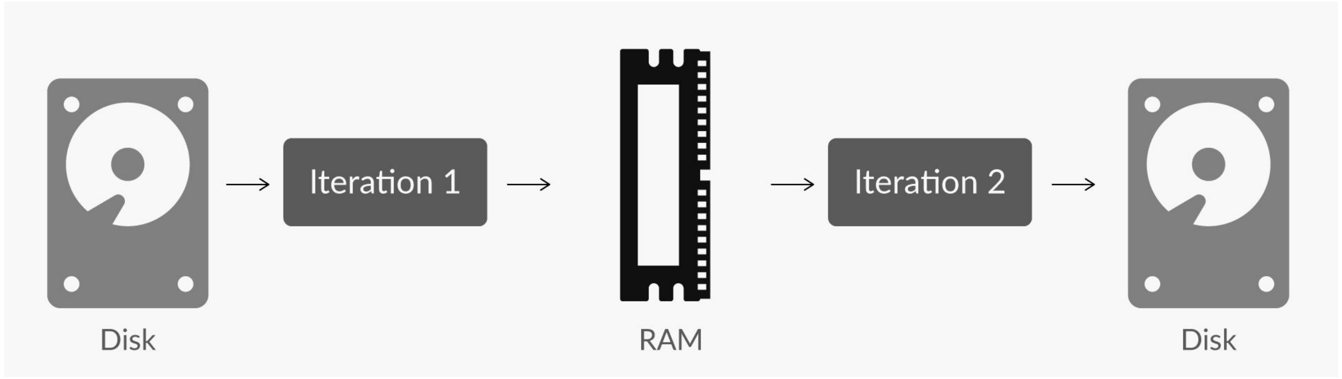
Map-Reduce vs. Spark: Iterative Processing

upGrad

Map-Reduce



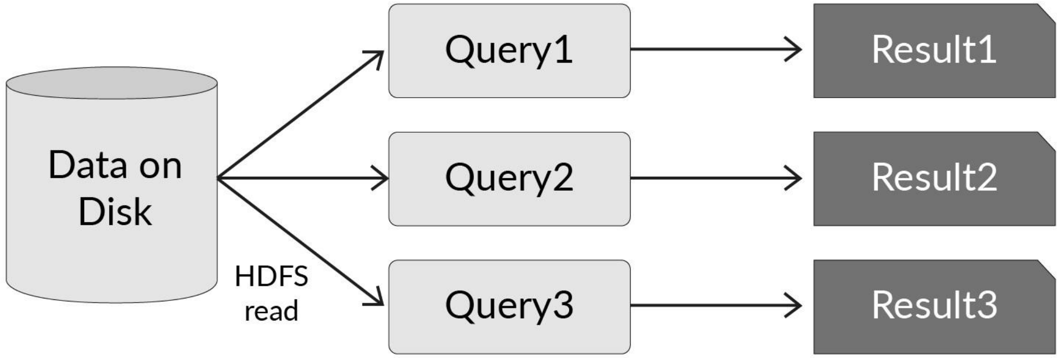
Spark



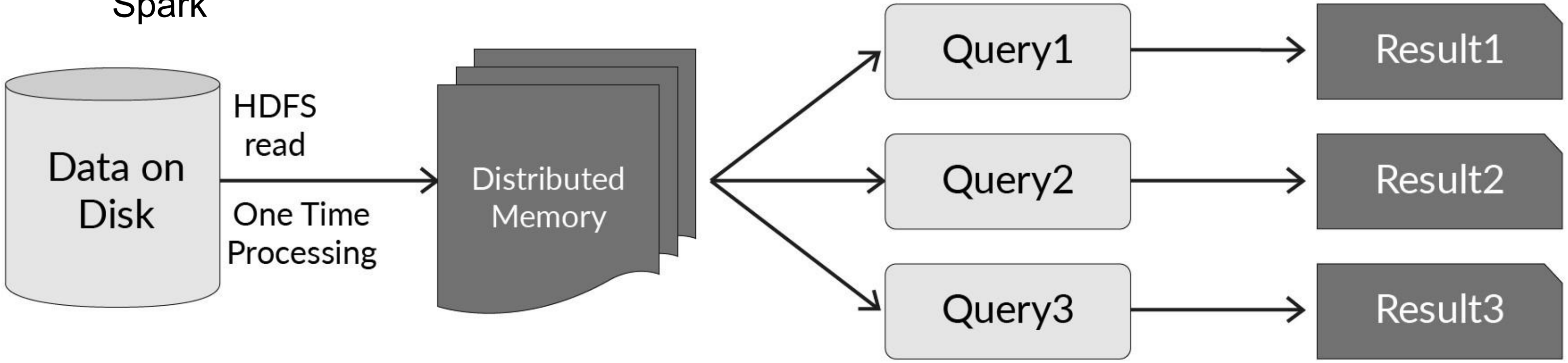
Map-Reduce vs. Spark: Interactive Processing

upGrad

Map-Reduce



Spark

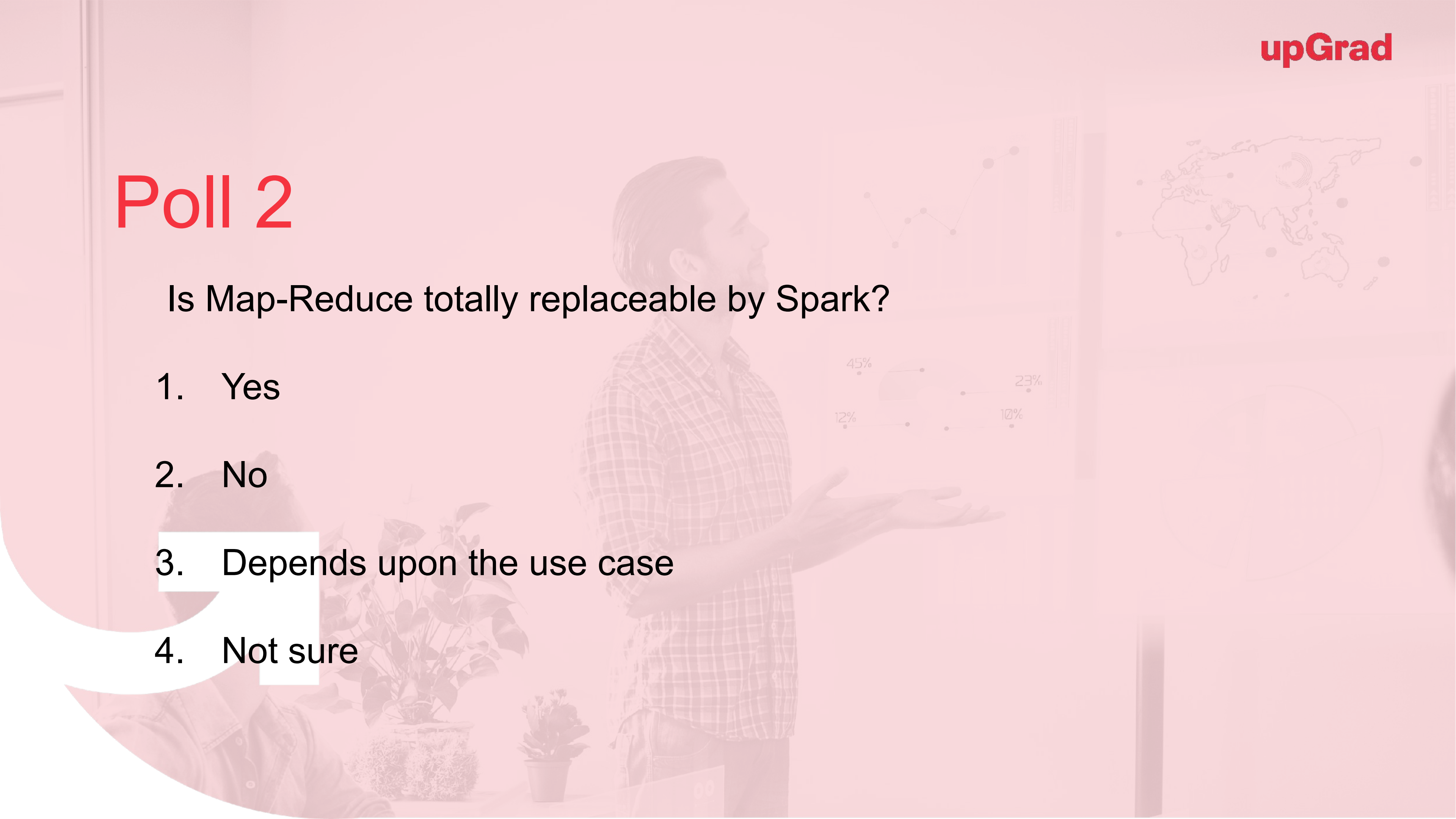


Map-Reduce vs. Spark

upGrad

Map Reduce	Spark
Well suited for batch jobs	Batch, stream and interactive processing
Disk based processing hence slow processing	Memory and disk based processing hence processing is fast
Support HDFS as data storage	Can run anywhere, local file system and or distributed file system, S3, GCP
Map and Reduce are the core processing framework	RDD is heart of Spark
Originally developed in **Java** but also support C++, Ruby, Groovy, Perl using Hadoop Streaming library	Originally developed in **Scala** but it Java also support out of box. It does support Java, Python, R and Sql to the full extent

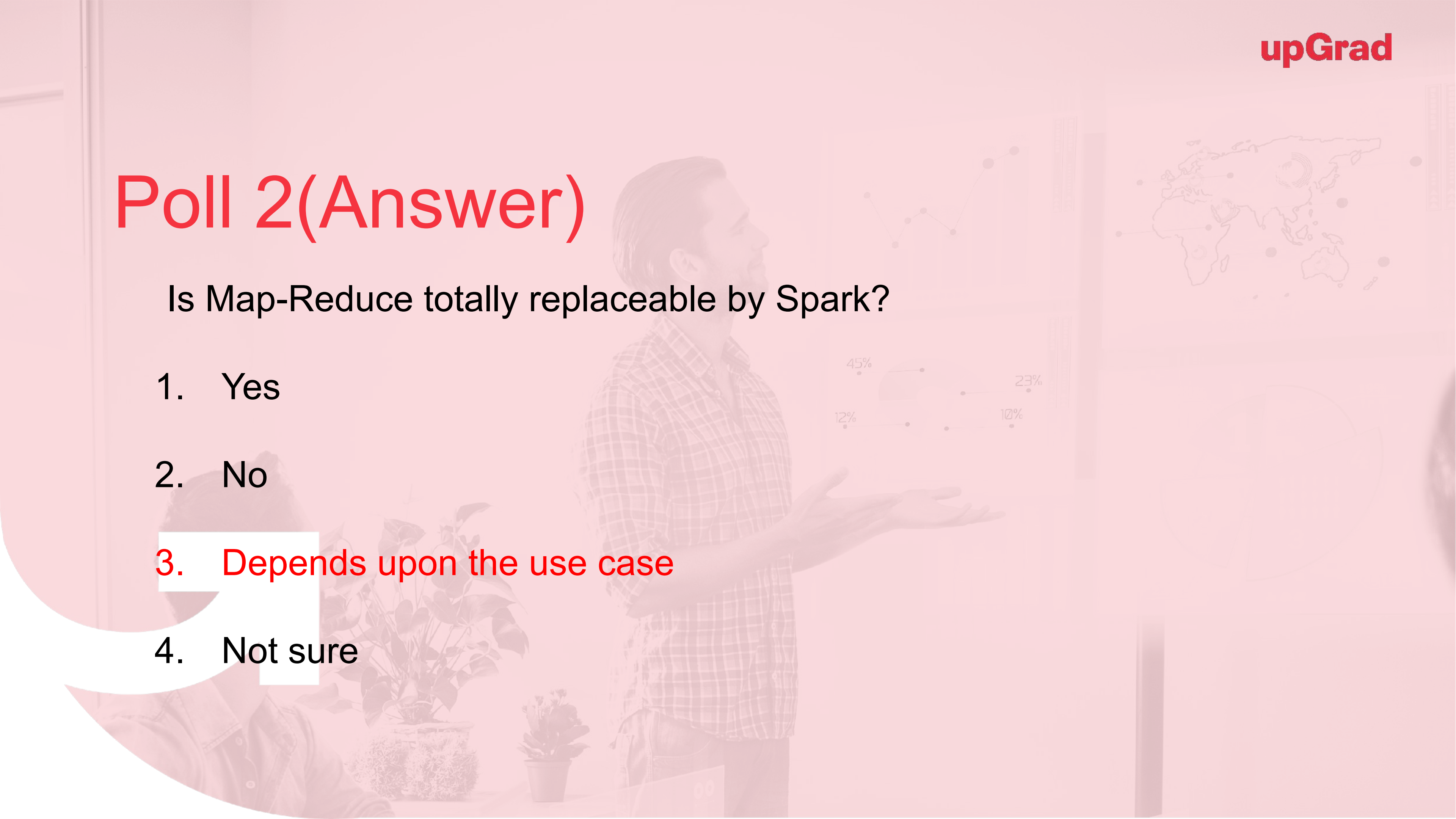
upGrad



Poll 2

Is Map-Reduce totally replaceable by Spark?

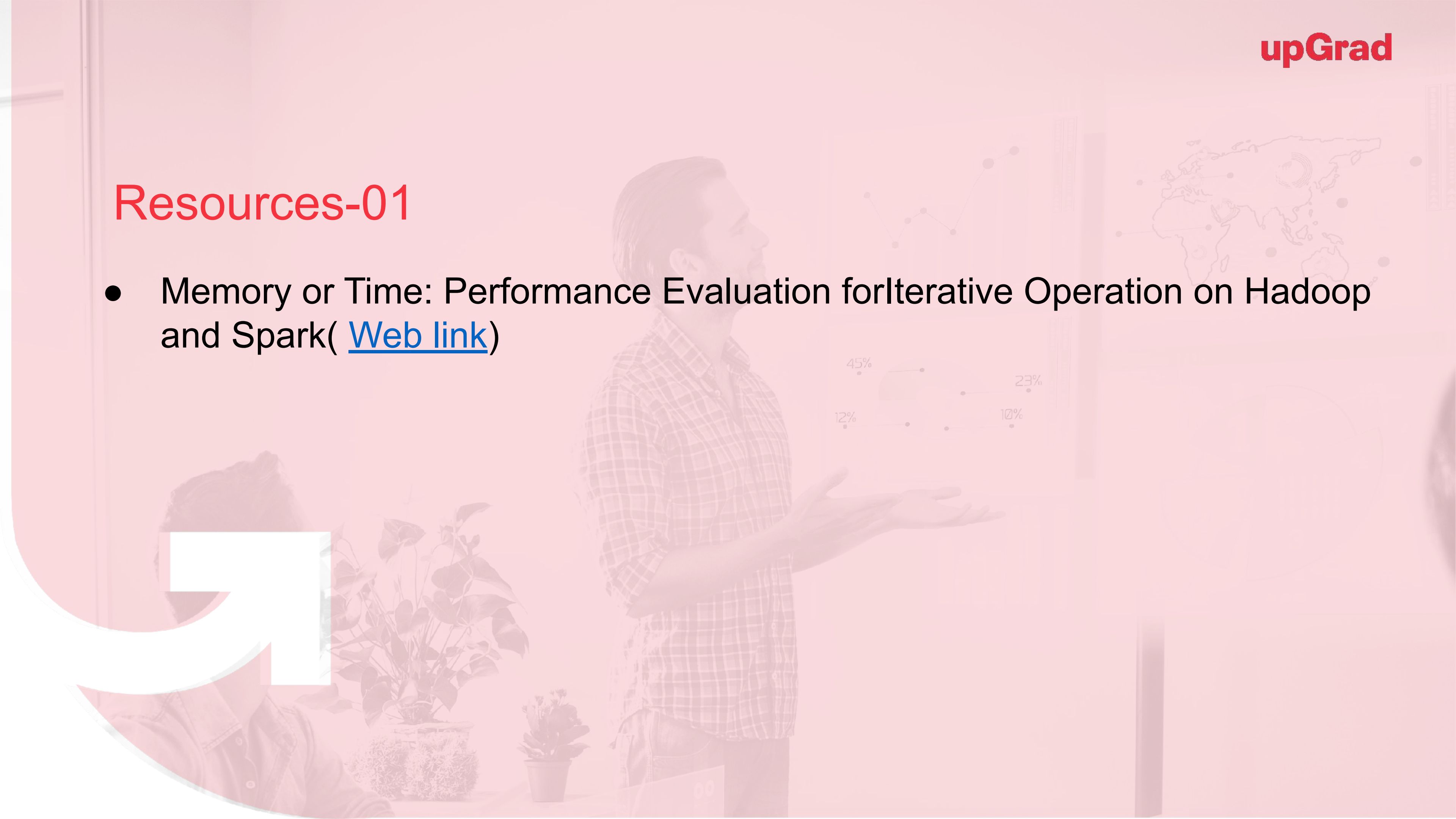
1. Yes
2. No
3. Depends upon the use case
4. Not sure



Poll 2(Answer)

Is Map-Reduce totally replaceable by Spark?

1. Yes
2. No
3. Depends upon the use case
4. Not sure



upGrad

Resources-01

- Memory or Time: Performance Evaluation forIterative Operation on Hadoop and Spark(Web link)

Spark RDDs

- A distributed memory abstraction that lets programmers perform in-memory computations on large clusters, in a fault-tolerant manner.

- The term 'RDD' is expanded as Resilient Distributed Dataset, where 'Resilient' refers to fault-tolerant, 'Distributed' refers to data that resides across multiple interconnected nodes, and 'Dataset' is a collection of partitioned data.

- RDDs are Spark Core’s abstraction for working with data. In other words, an RDD is a core object that is often used to deal with data in Spark.

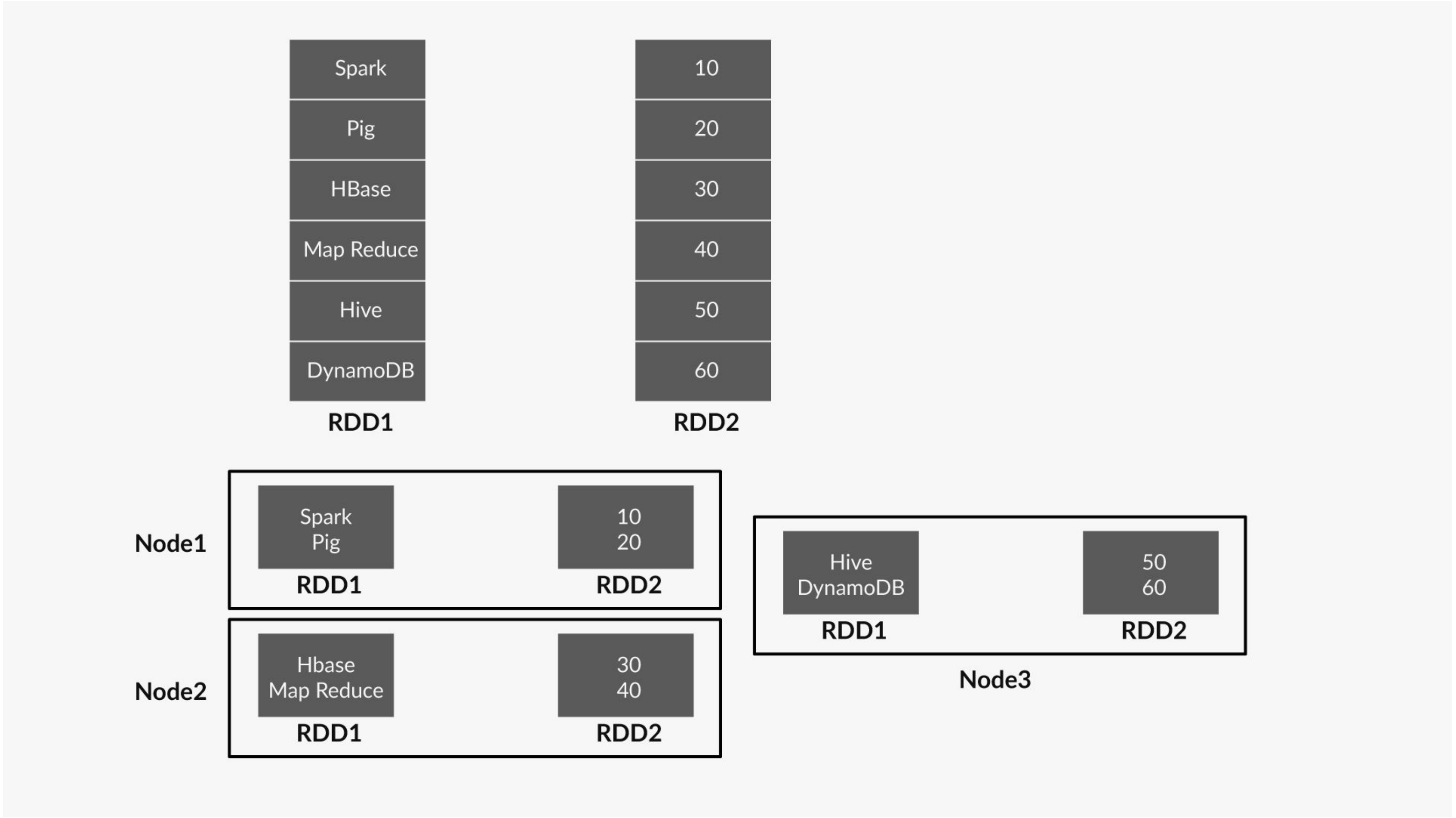
- An RDD is a fundamental data structure of Spark that is immutable and read-only.

- RDDs can include any Python, Java, or Scala objects, or even user-defined classes.

- Each dataset is divided into logical partitions that may be computed on different nodes of a cluster.

Pain RDD

upGrad



Pair RDDs

upGrad

Spark
Pig
HBase
Map Reduce
Hive
DynamoDB

RDD

India	200
Sri Lanka	100
Australia	80
Bangladesh	75
West Indies	60
South Africa	50

Paired RDD

Spark SQL

- RDDs are not the only way by which users can store and process data using Spark.
- Data Analysts who come from a non-programming background and are proficient in SQL can also work with Spark using Spark SQL.
- Spark SQL provides operators similar to SQL such as join, order by, group by, etc. using which analysts can get their work done quite easily avoiding the burden of writing long codes.
- Spark SQL introduced two data structures which can store the distributed data along with a schema. They are:
  - Dataframe
  - Dataset
- Dataframe and Datasets are built on top RDDs
- Dataframe and Datasets imposes schema on RDDs which are originally schemaless.

Spark Dataframe and Datasets

Dataframe:

- The first Dataframe API was introduced in Spark 1.3.
- It is an abstraction of a relational table in which the data is organised as named columns.
- A Dataframe**(Spark 2.x onwards)** is represented by a Dataset of type "Row" where "Row" is an untyped generic JVM object. i.e. **Dataset<Row> myDataframe**
- Dataframe APIs are **not** completely type-safe.
- It also introduced the Domain Specific Language API which provides SQL like operators such as group by, order by, join, etc. for easy manipulation of structured datasets.

Dataset:

- API was first introduced in Spark 1.6.
- On contrary to a Dataframe, a Dataset is a collection of strongly typed JVM objects such as String, Integers or of a Java bean type. i.e. **Dataset<Integer> myDataset**
- Being a collection of typed Java objects, Dataset API gives compile time error when a non-existing column is invoked. Which means Dataset API has type safety features.

upGrad

Opinion-01

Why is Spark RDDs are immutable?

Opinion-01(Answer)

Why is Spark RDDs are immutable?

- In a distributed parallel processing environment immutability of Spark RDD rules out the possibility of inconsistent results. In other words, Immutability solves the problems of problems due to concurrent use of the data set by multiple threads at once.
- Immutable data can as easily live in memory as on disk in a multiprocessing environment.
- Immutability of Spark RDDs also makes them a deterministic function of their input. Which means RDD can be recreated at any time. This helps in making RDDs fault tolerance.

Spark Architecture: Components

Spark Driver / Master Node

- The driver program runs on the master node.
- The driver is responsible for creating a SparkContext. However big or small the Spark job, a SparkContext is mandatory to run it.
- The driver monitors the end to end execution of a Spark program.

Spark Slave / Worker Nodes

- Slaves are the nodes on which executors run.
- Executors are processes that run smaller tasks that are assigned by the driver.
- They store computed results either in memory, cache or on hard disks.

Cluster Manager

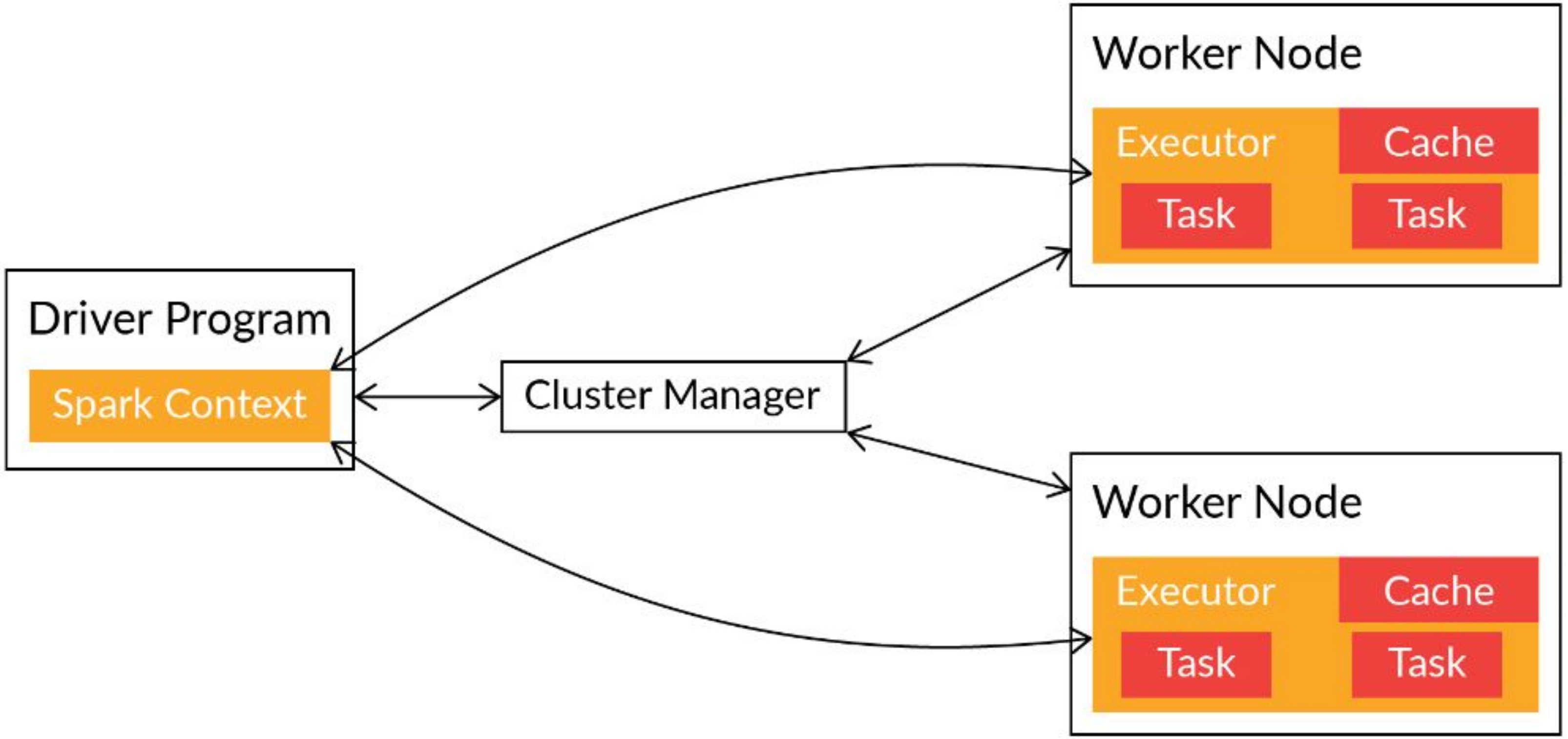
- The cluster manager is responsible for providing the resources required to run tasks on the cluster.
- The driver program negotiates with the cluster manager to get the required resources, such as the number of executors, the CPU, the memory, etc.

Spark Job deployment modes

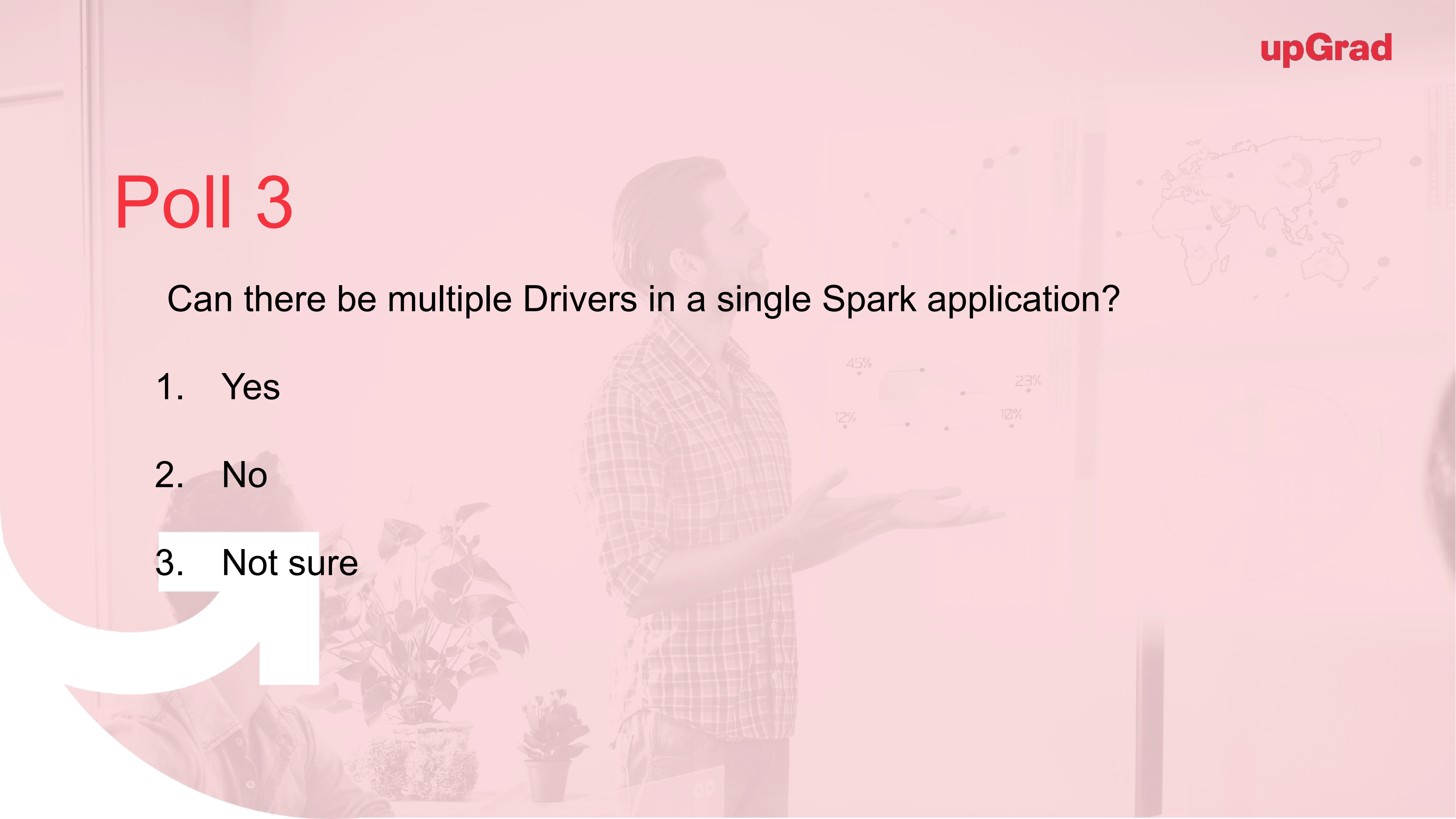
- **Local Mode**
- **Client Mode**
- **Cluster Mode**

Spark Architecture

upGrad



upGrad



Poll 3

Can there be multiple Drivers in a single Spark application?

1. Yes
2. No
3. Not sure

Poll 3(Answer)

Can there be multiple Drivers in a single Spark application?

1. Yes
2. [✓ No]
3. Not sure

Spark: Local Mode

upGrad

- Local mode is Spark is a non-distributed single-JVM deployment mode.

- In this mode, parallelism is specified using `setMaster("local[n]")` function while initializing the SparkSession or SparkContext object. Here, n is the number of CPU cores to be used during the execution.

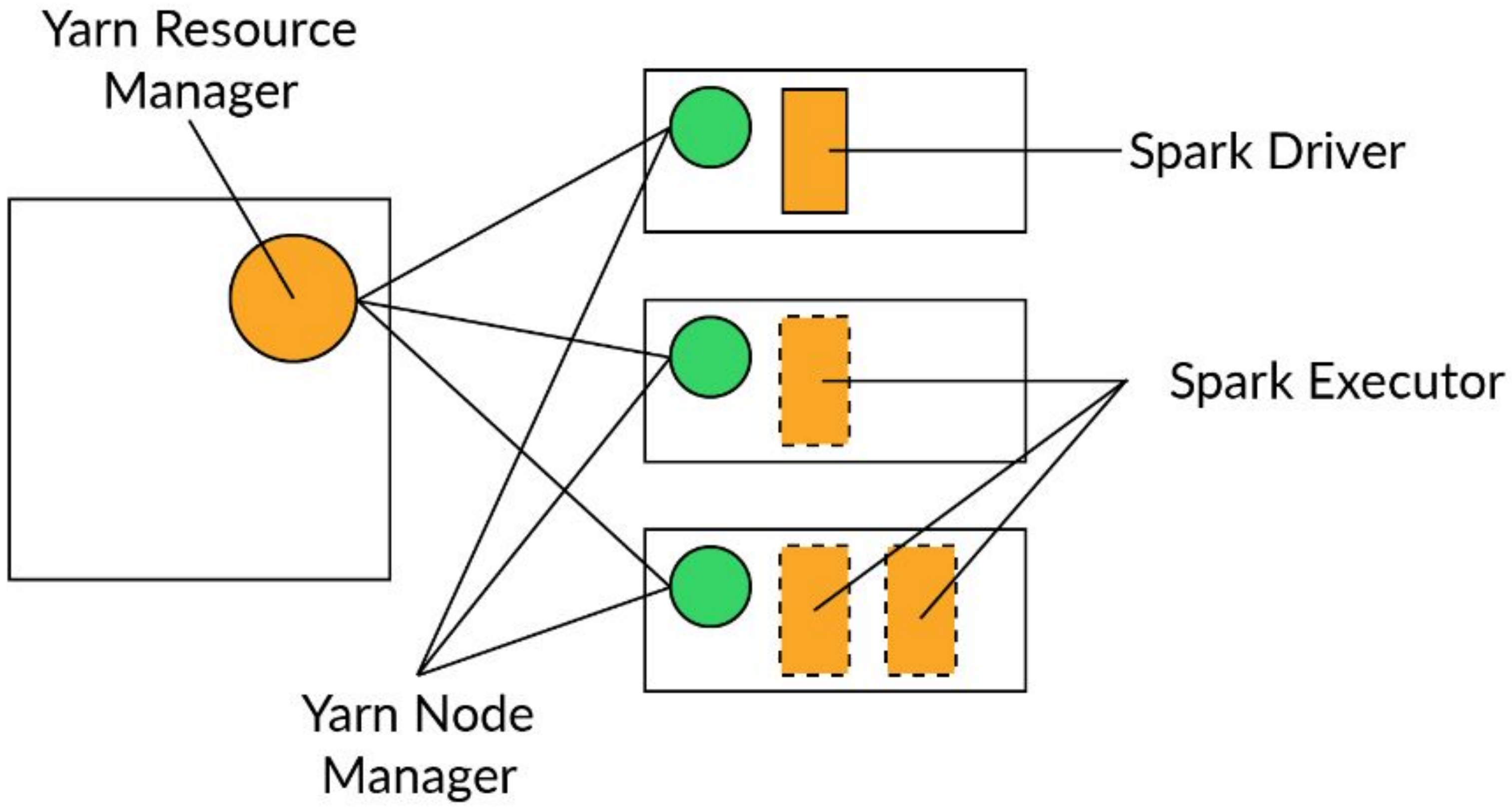
- This mode is very convenient for testing, debugging or demonstration purposes as it requires no previous setup to launch Spark applications.

- This mode is specified while initializing the Spark configuration object.

```
SparkConf conf = new SparkConf().setAppName("Spark_App").setMaster("local[2]");
```

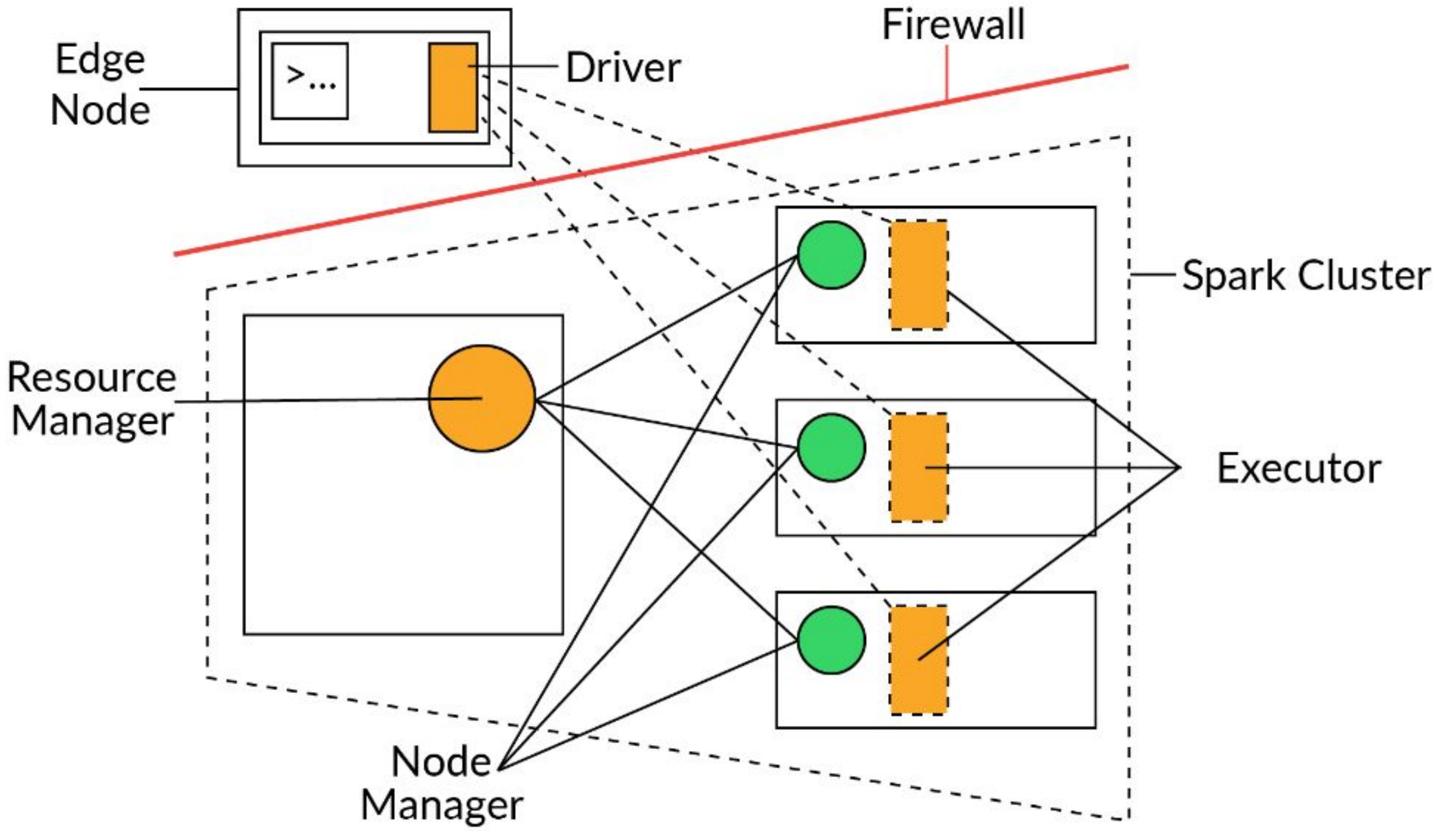
Spark: Cluster Mode

upGrad



Spark: Client Mode

upGrad



Spark: Job Deployment: Local

upGrad

- **Using master("local[n]") method in Eclipse**

```
SparkConf conf = new SparkConf().setAppName("Spark_App").setMaster("local[2]");
```

- **On EC2 instance**

```
# Run application locally on 2 cores
spark2-submit \
--class solution.Driver \
--master local[2] \
 /path/to/examples.jar \
 100
```

Spark: Job Deployment: Standalone

upGrad

```
# Run on a Spark standalone cluster in client deploy mode
spark2-submit \
  --class solution.Driver \
  --master spark://207.184.161.138:7077 \
  --deploy-mode client \
  --executor-memory 20G \
  --total-executor-cores 100 \
  /path/to/examples.jar \
  1000

# Run on a Spark standalone cluster in cluster deploy mode
spark2-submit \
  --class solution.Driver \
  --master spark://207.184.161.138:7077 \
  --deploy-mode cluster \
  --executor-memory 20G \
  --total-executor-cores 100 \
  /path/to/examples.jar \
  1000
```

Spark: Job Deployment: YARN

upGrad

```
# Run on a Spark Hadoop cluster in Yarn Client deploy mode
spark2-submit \
--class solution.Driver \
--master yarn\
--deploy-mode client \
/path/to/examples.jar \
  1000

# Run on a Spark Hadoop cluster in Yarn Cluster deploy mode
spark2-submit \
--class solution.Driver \
--master yarn\
--deploy-mode cluster \
--executor-memory 20G \
--driver-memory 2G \
--total-executor-cores 100 \
/path/to/examples.jar \
1000
```

Transformations and Actions

upGrad

- Spark allows two types of operations on RDDs-
  - **Transformations:** Operation on an RDD returns another RDD.
  - **Actions:** Operation on an RDD returns a non-RDD value.

- **Some common Transformations are-**
  - **map**(): There is a one-to-one mapping between the elements of the source RDD and those of the resultant RDD.
  - **filter()**: used to filter the data
  - **flatMap()**: The flatMap transformation maps an element of the input RDD to one or more elements in the target RDD.
  - **mapToPair()**: Creates one to one mapping between the elements of source Pair RDD and the resultant Pari RDD.
  - **flatMapToPair()**: It is similar to flatMap() with a difference that the target RDD produced is a paired RDD.
  - **reduceByKey**: used to transform a paired RDD by aggregating elements based on the common key.

Transformation and Actions

- **Some common Actions are**
  - **collect:** Collects all the data for a particular RDD distributed across partitions and returns it to the driver program.
  - **count**: Returns the total number of elements that exist in RDDs.
  - **first**: First returns the first element of an RDD.
  - **take:** The take action takes an integer parameter "n" and returns a list of first "n" elements of the RDD.
  - **isEmpty**: Returns a boolean value indicating whether the given RDD is empty or not
  - **reduce**: The reduce action takes a lambda expression as an input and aggregates the elements of the input RDD.
  - **saveAsTextFile:** It is useful for storing all the data items present in the RDD in persistent storage such as a local file system or HDFS.

Program Execution: Lazy Evaluation

upGrad

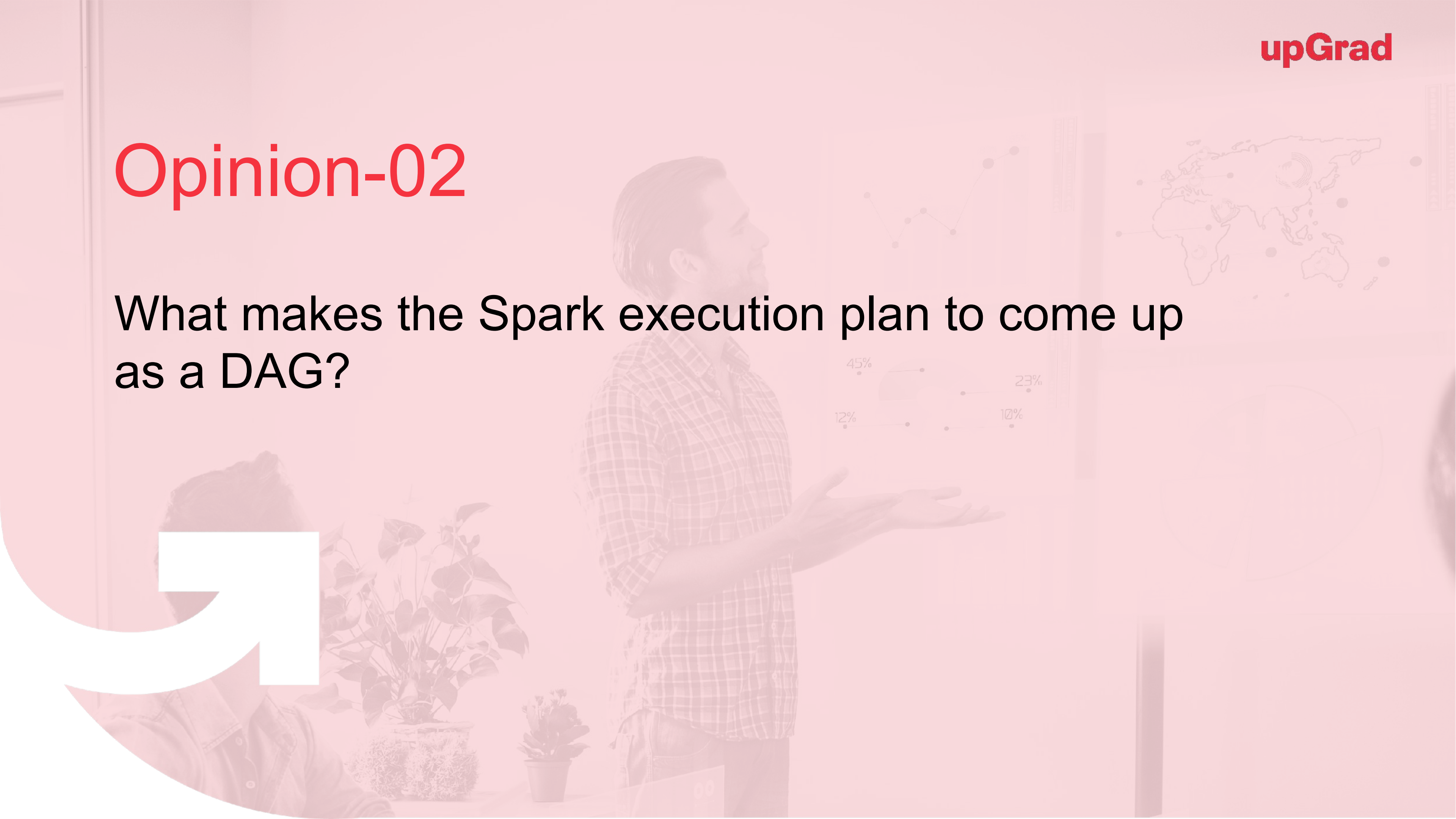
- Spark evaluates something only when it is needed.
- The transformations in Spark are not performed as soon as they are encountered.
- All the transformations performed on an RDD are stored in a Directed Acyclic Graph aka DAG.
- The DAG is executed only when an action is performed on an RDD. This is known as lazy evaluation in Spark.

- **Some benefits of lazy evaluation are-**
  - **Reduction in complexities:** Lazy evaluation avoids execution of all the operations. Internally the operations are optimised and the number of executable operations is reduced. This reduction in operations generates less intermediate data, and the task executes faster.
  - **Saves computation and increases speed:** As all the operations are not performed, and only those operations or values are computed which are required, this reduces the time and space complexity.
  - **Fault tolerance:** Due to lineage graphs and lazy evaluation, Spark is well aware of the steps in advance to create an RDD. So, if a partition of an RDD is lost, it can be efficiently computed from the previous level in DAG without performing all the steps

upGrad

Opinion-02

What makes the Spark execution plan to come up as a DAG?



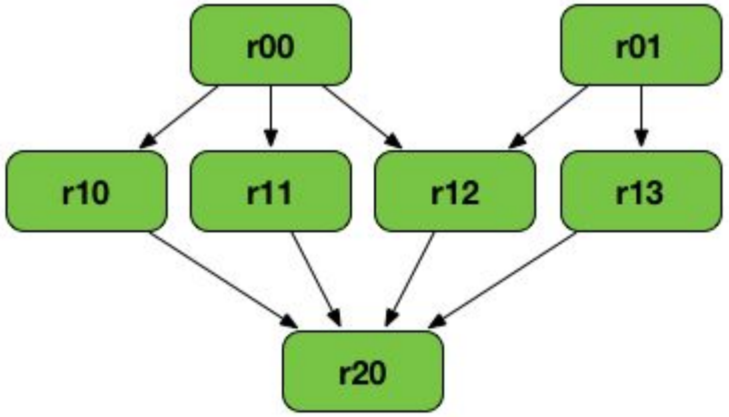
upGrad

Opinion-02(Answer)

What makes the Spark execution plan to come up as a DAG?

- Because of the Immutability of Spark RDDs every time we apply a transformation a new RDD is created, that means a new node is added to the lineage graph, which builds the entire execution plan as a DAG.

Lazy Evaluation: Lineage Graph

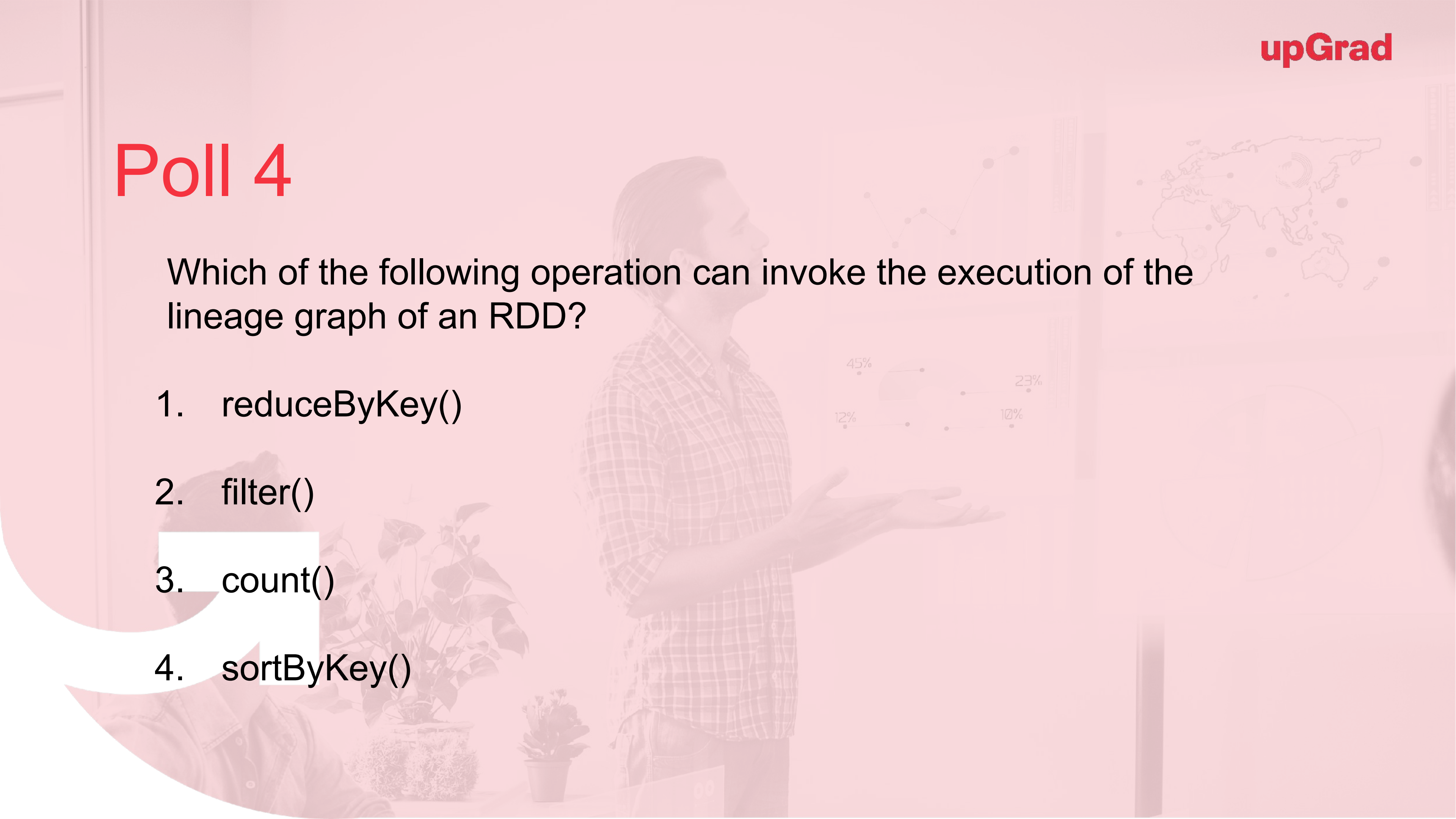


upGrad

Poll 4

Which of the following operation can invoke the execution of the lineage graph of an RDD?

1. reduceByKey()
2. filter()
3. count()
4. sortByKey()



upGrad

Poll 4(Answer)

Which of the following operation can invoke the execution of the lineage graph of an RDD?

1. reduceByKey()
2. filter()
3. count()
4. sortByKey()

Advance Topics

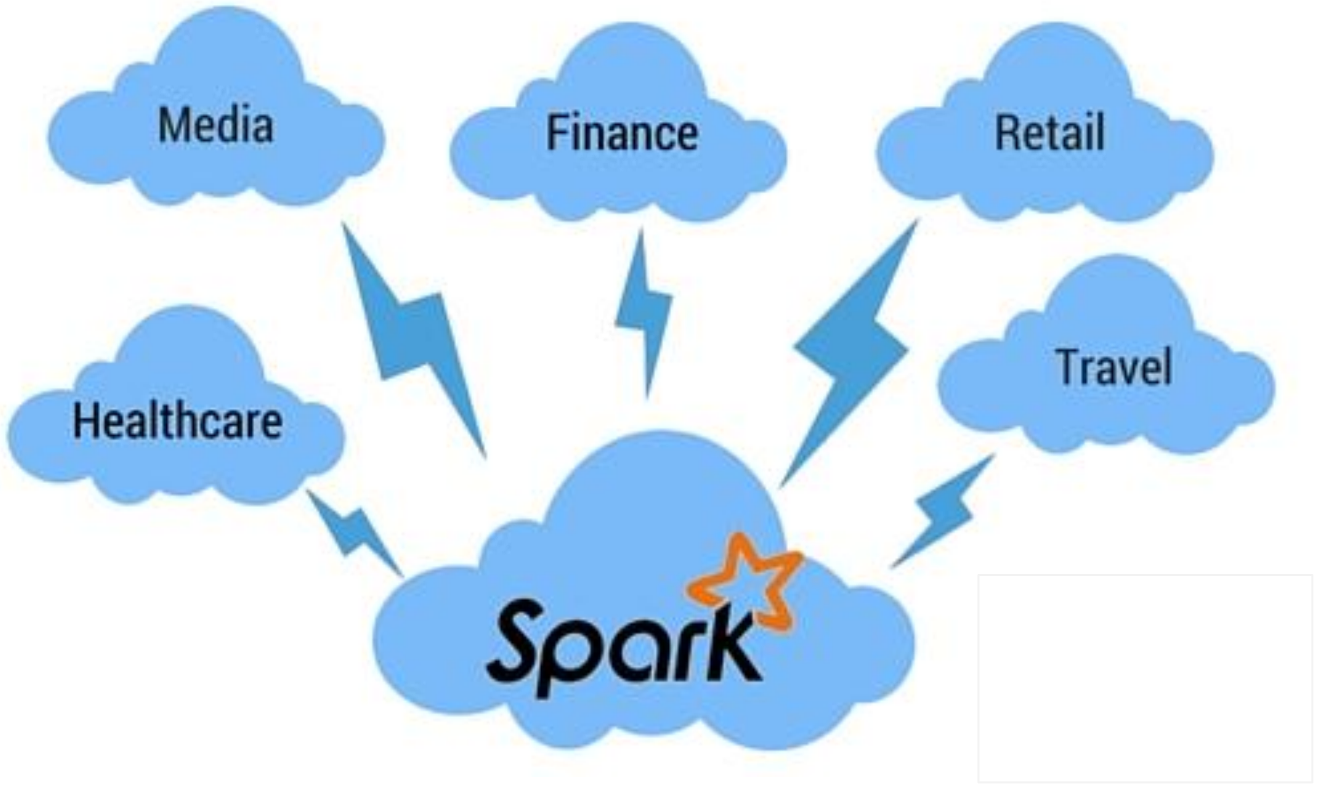
upGrad

- Shared Variables in Spark
  - **Broadcast Variable**
    - Broadcast variables are read-only variables.
    - A broadcast variable is serialised and sent to each container where it gets cached, this avoids shipping a copy with each task.
  - **Accumulator Variable**
    - An accumulator is a shared variable which can be incremented(addition).
    - This is similar to the Counters in MapReduce.

- Data Partitioning in Spark
  - **Hash Partitioning**
    - Hash Partitioning calculates the Hash of key to deciding the partition for a key-value pair.
  - **Range Partitioning**
    - In the range partitioning method, tuples are first sorted and then, partitioned based on the particular range of keys

Spark in the Industry

upGrad



Limitation of Spark

upGrad

Limitations of Apache Spark:

- **Doesn't have a native file management system:** Apache Spark relies on other platforms like Hadoop, S3 or some another cloud-based Platform for the file management system.
- **No support for Real-Time Processing**: With Spark Streaming it can only support near real-time processing of live data.
- **Manual Optimization**: Manual Optimization is required to optimize Spark jobs. Also, it is adequate to specific datasets. we need to control manually if we want to partition and cache in Spark to be correct.
- **Less no. of Algorithm:** Spark MLlib lags behind in terms of a number of available algorithms like Tanimoto distance.
- **Window Criteria**: Spark does not support record based window criteria. It only has time-based window criteria.
- **Expensive**: The lightning fast in-memory processing capability also adds up to a higher cost.

Code Example

upGrad

- **Practice Problem: Q2:**
  - Year wise get the count of top 4 Airports with the maximum number of outgoing flights, order the results in descending order w.r.t the number of outgoing flights.

- Using
  - Spark RDDs
  - Spark Dataframe
  - Spark SQL
  - Spark RDDs in Scala

upGrad

Q & A

upGrad

Key Takeaway

- What is Apache Spark
- Spark Architecture
- How to write a Spark Java program

upGrad

Thank You!