# Spark Interview Question

## 1. How is Apache Spark different from MapReduce?

| Apache Spark | MapReduce |
|---|---|
| Spark processes data in batches as well as in real-time | MapReduce processes data in batches only |
| Spark runs almost 100 times faster than Hadoop MapReduce | Hadoop MapReduce is slower when it comes to large scale data processing |
| Spark stores data in the RAM i.e. in-memory. So, it is easier to retrieve it | Hadoop MapReduce data is stored in HDFS and hence takes a long time to retrieve the data |
| Spark provides caching and in-memory data storage | Hadoop is highly disk-dependent |

## 2. What are the important components of the Spark ecosystem?

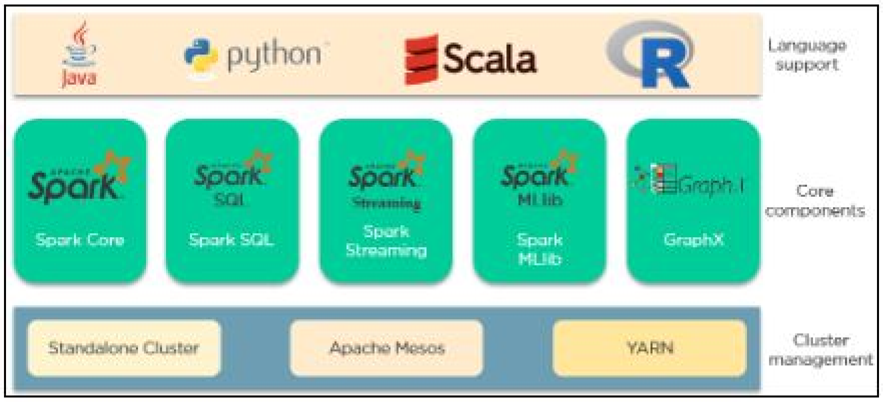


Figure 1: https://www.simplilearn.com/ice9/free_resources_article_thumb/apache-spark.JPG

Apache Spark has 3 main categories that comprise its ecosystem. Those are:

- Language support: Spark can integrate with different languages to applications and perform analytics. These languages are Java, Python, Scala, and R.
- Core Components: Spark supports 5 main core components. There are Spark Core, Spark SQL, Spark Streaming, Spark MLlib, and GraphX.
- Cluster Management: Spark can be run in 3 environments. Those are the Standalone cluster, Apache Mesos, and YARN.

## 3. Explain how Spark runs applications with the help of its architecture.

This is one of the most frequently asked spark interview questions, and the interviewer will expect you to give a thorough answer to it.

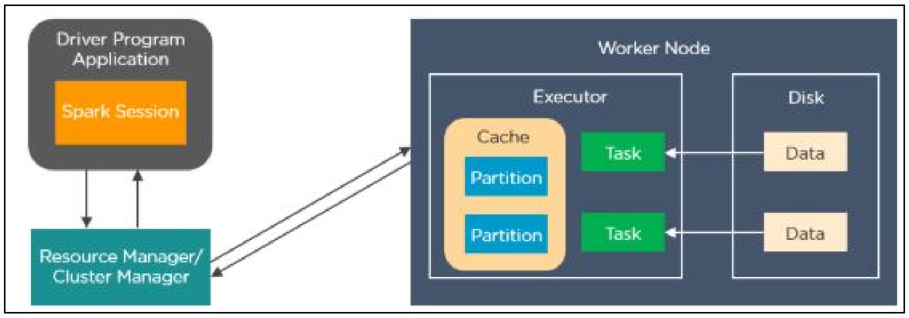


Figure 2: https://www.simplilearn.com/ice9/free_resources_article_thumb/worker-node.JPG

Spark applications run as independent processes that are coordinated by the SparkSession object in the driver program. The resource manager or cluster manager assigns tasks to the worker nodes with one task per partition. Iterative algorithms apply operations repeatedly to the data so they can benefit from caching datasets across iterations. A task applies its unit of work to the dataset in its partition and outputs a new partition dataset. Finally, the results are sent back to the driver application or can be saved to the disk.

## 4. What are the different cluster managers available in Apache Spark?

- Standalone Mode: By default, applications submitted to the standalone mode cluster will run in FIFO order, and each application will try to use all available nodes. You can launch a standalone cluster either manually, by starting a master and workers by hand, or use our provided launch scripts. It is also possible to run these daemons on a single machine for testing.
- Apache Mesos: Apache Mesos is an open-source project to manage computer clusters, and can also run Hadoop applications. The advantages of deploying Spark with Mesos include dynamic partitioning between Spark and other frameworks as well as scalable partitioning between multiple instances of Spark.
- Hadoop YARN: Apache YARN is the cluster resource manager of Hadoop 2. Spark can be run on YARN as well.
- Kubernetes: Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

## 5. What is the significance of Resilient Distributed Datasets in Spark?

Resilient Distributed Datasets are the fundamental data structure of Apache Spark. It is embedded in Spark Core. RDDs are immutable, fault-tolerant, distributed collections of objects that can be operated on in parallel. RDD's are split into partitions and can be executed on different nodes of a cluster.

RDDs are created by either transformation of existing RDDs or by loading an external dataset from stable storage like HDFS or HBase.

Here is how the architecture of RDD looks like:

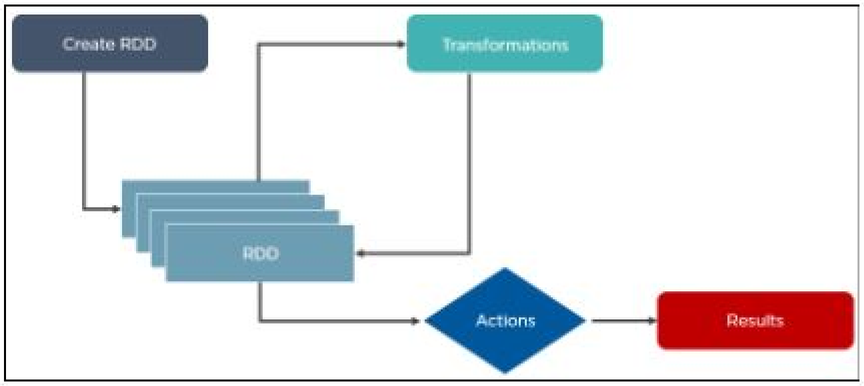


Figure 3: https://www.simplilearn.com/ice9/free_resources_article_thumb/create-rdd.JPG

So far, if you have any doubts regarding the apache spark interview questions and answers, please comment below.

## 6. What is a lazy evaluation in Spark?

When Spark operates on any dataset, it remembers the instructions. When a transformation such as a map() is called on an RDD, the operation is not performed instantly. Transformations in Spark are not evaluated until you perform an action, which aids in optimizing the overall data processing workflow, known as lazy evaluation.

## 7. What makes Spark good at low latency workloads like graph processing and Machine Learning?

Apache Spark stores data in-memory for faster processing and building machine learning models. Machine Learning algorithms require multiple iterations and different conceptual steps to create an optimal model. Graph algorithms traverse through all the nodes and edges to generate a graph. These low latency workloads that need multiple iterations can lead to increased performance.

### 8. How can you trigger automatic clean-ups in Spark to handle accumulated metadata?

To trigger the clean-ups, you need to set the parameter spark.cleaner.ttlx.

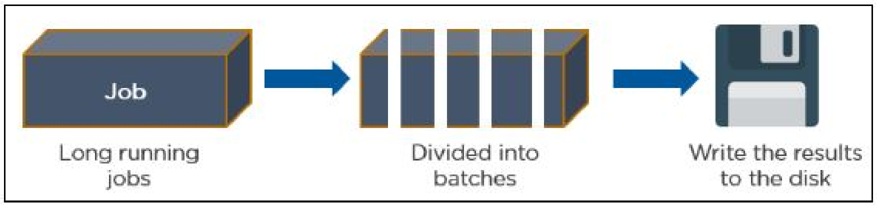


Figure 4: https://www.simplilearn.com/ice9/free_resources_article_thumb/job.JPG

### 9. How can you connect Spark to Apache Mesos?

There are a total of 4 steps that can help you connect Spark to Apache Mesos.

- Configure the Spark Driver program to connect with Apache Mesos
- Put the Spark binary package in a location accessible by Mesos
- Install Spark in the same location as that of the Apache Mesos
- Configure the spark.mesos.executor.home property for pointing to the location where Spark is installed

### 10. What is a Parquet file and what are its advantages?

Parquet is a columnar format that is supported by several data processing systems. With the Parquet file, Spark can perform both read and write operations.

Some of the advantages of having a Parquet file are:

- It enables you to fetch specific columns for access.
- It consumes less space
- It follows the type-specific encoding
- It supports limited I/O operations

### 11. What is shuffling in Spark? When does it occur?

Shuffling is the process of redistributing data across partitions that may lead to data movement across the executors. The shuffle operation is implemented differently in Spark compared to Hadoop.

Shuffling has 2 important compression parameters:

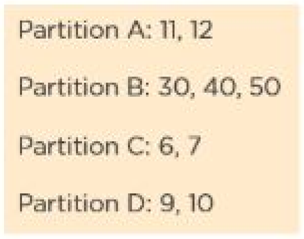
spark.shuffle.compress – checks whether the engine would compress shuffle outputs or not spark.shuffle.spill.compress – decides whether to compress intermediate shuffle spill files or not

It occurs while joining two tables or while performing byKey operations such as GroupByKey or ReduceByKey

**12. What is the use of coalesce in Spark?**

Spark uses a coalesce method to reduce the number of partitions in a DataFrame.

Suppose you want to read data from a CSV file into an RDD having four partitions.



Partition A: 11, 12

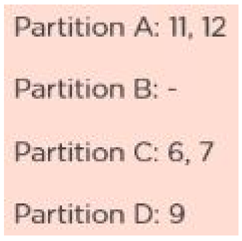
Partition B: 30, 40, 50

Partition C: 6, 7

Partition D: 9, 10

Figure 5: https://www.simplilearn.com/ice9/free_resources_article_thumb/partition.JPG

This is how a filter operation is performed to remove all the multiple of 10 from the data.



Partition A: 11, 12

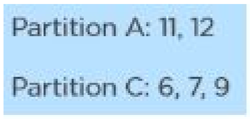
Partition B: -

Partition C: 6, 7

Partition D: 9

Figure 6: https://www.simplilearn.com/ice9/free_resources_article_thumb/partition-2.JPG

The RDD has some empty partitions. It makes sense to reduce the number of partitions, which can be achieved by using coalesce.



Partition A: 11, 12

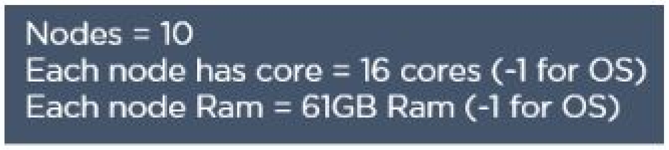
Partition C: 6, 7, 9

Figure 7: https://www.simplilearn.com/ice9/free_resources_article_thumb/partition-3.JPG

This is how the resultant RDD would look like after applying to coalesce.

**13. How can you calculate the executor memory?**

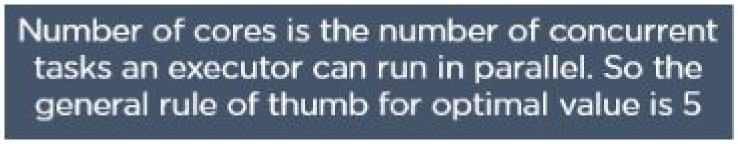
Consider the following cluster information:



Nodes = 10
Each node has core = 16 cores (-1 for OS)
Each node Ram = 61GB Ram (-1 for OS)

Figure 8: https://www.simplilearn.com/ice9/free_resources_article_thumb/cluster-new.JPG

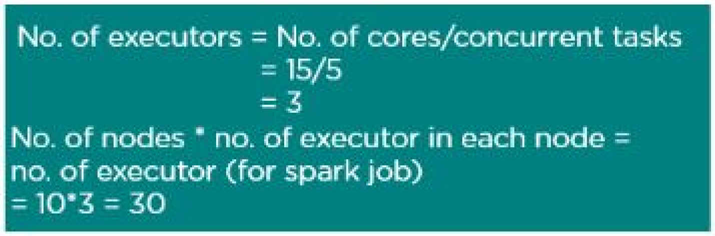
Here is the number of core identification:



Number of cores is the number of concurrent tasks an executor can run in parallel. So the general rule of thumb for optimal value is 5

Figure 9: https://www.simplilearn.com/ice9/free_resources_article_thumb/core-iden.JPG

To calculate the number of executor identification:



```
No. of executors = No. of cores/concurrent tasks
                 = 15/5
                 = 3
No. of nodes * no. of executor in each node =
no. of executor (for spark job)
= 10*3 = 30
```

Figure 10: https://www.simplilearn.com/ice9/free_resources_article_thumb/executor.JPG

**14. What are the various functionalities supported by Spark Core?**

Spark Core is the engine for parallel and distributed processing of large data sets. The various functionalities supported by Spark Core include:

- Scheduling and monitoring jobs
- Memory management
- Fault recovery
- Task dispatching

**15. How do you convert a Spark RDD into a DataFrame?**

There are 2 ways to convert a Spark RDD into a DataFrame:

- Using the helper function - toDF

import com.mapr.db.spark.sql._

val df = sc.loadFromMapRDB()

.where(field("first_name") === "Peter")

.select("_id", "first_name").toDF()

- Using SparkSession.createDataFrame

You can convert an RDD[Row] to a DataFrame by

calling createDataFrame on a SparkSession object

def createDataFrame(RDD, schema:StructType)

**16. Explain the types of operations supported by RDDs.**

Resilient Distributed Dataset (RDD) is a rudimentary data structure of Spark. RDDs are the immutable Distributed collections of objects of any type. It records the data from various nodes and prevents it from significant faults.

The Resilient Distributed Dataset (RDD) in Spark supports two types of operations. These are:

1. Transformations
2. Actions

### RDD Transformation:

The transformation function generates new RDD from the pre-existing RDDs in Spark. Whenever the transformation occurs, it generates a new RDD by taking an existing RDD as input and producing one or more RDD as output. Due to its Immutable nature, the input RDDs don't change and remain constant.

Along with this, if we apply Spark transformation, it builds RDD lineage, including all parent RDDs of the final RDDs. We can also call this RDD lineage as RDD operator graph or RDD dependency graph. RDD Transformation is the logically executed plan, which means it is a Directed Acyclic Graph (DAG) of the continuous parent RDDs of RDD.

### RDD Action:

The RDD Action works on an actual dataset by performing some specific actions. Whenever the action is triggered, the new RDD does not generate as happens in transformation. It depicts that Actions are Spark RDD operations that provide non-RDD values. The drivers and external storage systems store these non-RDD values of action. This brings all the RDDs into motion.

If appropriately defined, the action is how the data is sent from the Executor to the driver. Executors play the role of agents and the responsibility of executing a task. In comparison, the driver works as a JVM process facilitating the coordination of workers and task execution.

### 17. What is a Lineage Graph?

This is another frequently asked spark interview question. A Lineage Graph is a dependencies graph between the existing RDD and the new RDD. It means that all the dependencies between the RDD will be recorded in a graph, rather than the original data.

The need for an RDD lineage graph happens when we want to compute a new RDD or if we want to recover the lost data from the lost persisted RDD. Spark does not support data replication in memory. So, if any data is lost, it can be rebuilt using RDD lineage. It is also called an RDD operator graph or RDD dependency graph.

### 18. What do you understand about DStreams in Spark?

A Discretized Stream (DStream) is a continuous sequence of RDDs and the rudimentary abstraction in Spark Streaming. These RDDs sequences are of the same type representing a constant stream of data. Every RDD contains data from a specific interval.

The DStreams in Spark take input from many sources such as Kafka, Flume, Kinesis, or TCP sockets. It can also work as a data stream generated by converting the input stream. It facilitates developers with a high-level API and fault tolerance.

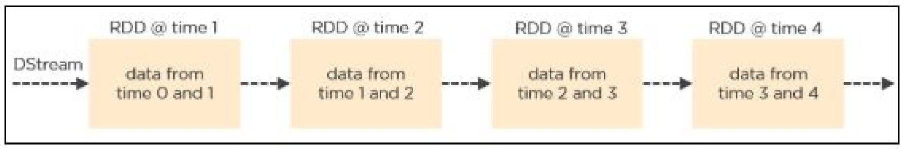


Figure 11: https://www.simplilearn.com/ice9/free_resources_article_thumb/dstream.JPG

### 19. Explain Caching in Spark Streaming.

Caching also known as Persistence is an optimization technique for Spark computations. Similar to RDDs, DStreams also allow developers to persist the stream's data in memory. That is, using the persist() method on a DStream will automatically persist every RDD of that DStream in memory. It helps to save interim partial results so they can be reused in subsequent stages.

The default persistence level is set to replicate the data to two nodes for fault-tolerance, and for input streams that receive data over the network.

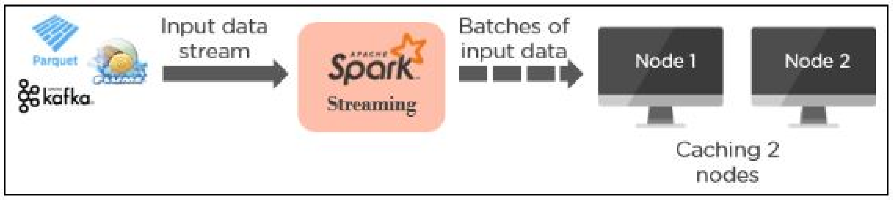


Figure 12: https://www.simplilearn.com/ice9/free_resources_article_thumb/kafka.JPG

## 20. What is the need for broadcast variables in Spark?

Broadcast variables allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks. They can be used to give every node a copy of a large input dataset in an efficient manner. Spark distributes broadcast variables using efficient broadcast algorithms to reduce communication costs.

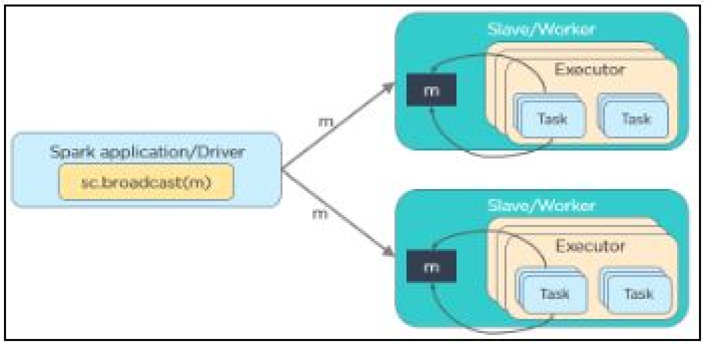


Figure 13: https://www.simplilearn.com/ice9/free_resources_article_thumb/scala.JPG

scala> val broadcastVar = sc.broadcast(Array(1, 2, 3))

broadcastVar: org.apache.spark.broadcast.Broadcast[Array[Int]] = Broadcast(0)

scala> broadcastVar.value

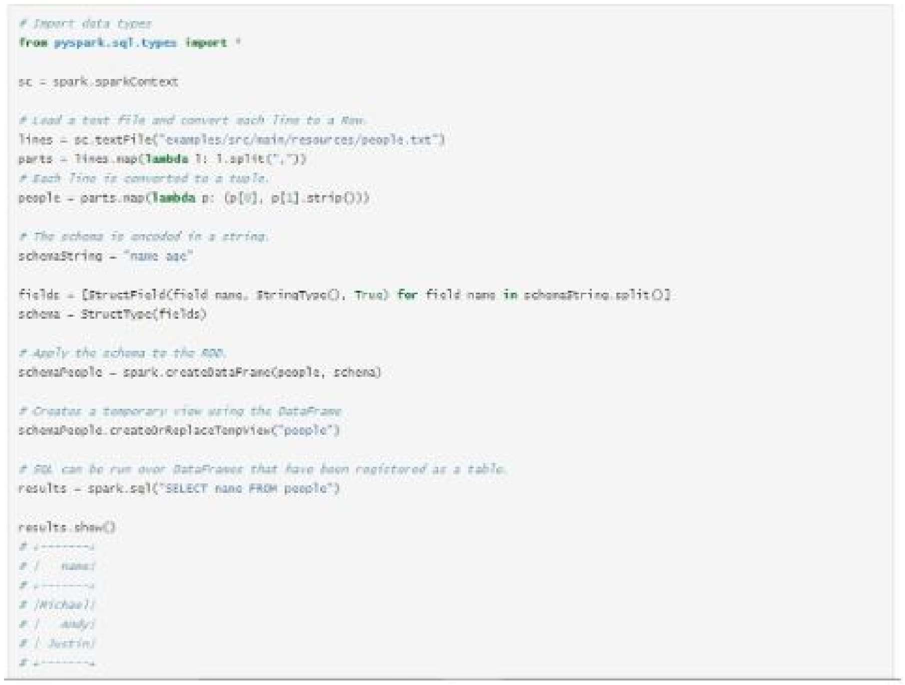
res0: Array[Int] = Array(1, 2, 3)

## Apache Spark Interview Questions for Experienced

### 21. How to programmatically specify a schema for DataFrame?

DataFrame can be created programmatically with three steps:

- Create an RDD of Rows from the original RDD;
- Create the schema represented by a StructType matching the structure of Rows in the RDD created in Step 1.
- Apply the schema to the RDD of Rows via createDataFrame method provided by SparkSession.



```
# Import data types
from pyspark.sql.types import *

sc = spark.sparkContext

# Load a text file and convert each line to a Row.
lines = sc.textFile("examples/src/main/resources/people.txt")
parts = lines.map(lambda l: l.split(","))
# Each line is converted to a tuple.
people = parts.map(lambda p: (p[0], p[1].strip()))

# The schema is encoded in a string.
schemaString = "name age"

fields = [StructField(field_name, StringType(), True) for field_name in schemaString.split()]
schema = StructType(fields)

# Apply the schema to the RDD.
schemaPeople = spark.createDataFrame(people, schema)

# Creates a temporary view using the DataFrame
schemaPeople.createOrReplaceTempView("people")

# SQL can be run over DataFrames that have been registered as a table.
results = spark.sql("SELECT name FROM people")

results.show()
# +-------+
# |   name|
# +-------+
# |Michael|
# |   Andy|
# | Justin|
# +-------+
```

Figure 14: https://www.simplilearn.com/ice9/free_resources_article_thumb/spark-session.JPG

### 22. Which transformation returns a new DStream by selecting only those records of the source DStream for which the function returns true?

1. map(func)
2. transform(func)
3. filter(func)

4. count()

The correct answer is c) filter(func).

## 23. Does Apache Spark provide checkpoints?

This is one of the most frequently asked spark interview questions where the interviewer expects a detailed answer (and not just a yes or no!). Give as detailed an answer as possible here.

Yes, Apache Spark provides an API for adding and managing checkpoints. Checkpointing is the process of making streaming applications resilient to failures. It allows you to save the data and metadata into a checkpointing directory. In case of a failure, the spark can recover this data and start from wherever it has stopped.

There are 2 types of data for which we can use checkpointing in Spark.

Metadata Checkpointing: Metadata means the data about data. It refers to saving the metadata to fault-tolerant storage like HDFS. Metadata includes configurations, DStream operations, and incomplete batches.

Data Checkpointing: Here, we save the RDD to reliable storage because its need arises in some of the stateful transformations. In this case, the upcoming RDD depends on the RDDs of previous batches.

## 24. What do you mean by sliding window operation?

Controlling the transmission of data packets between multiple computer networks is done by the sliding window. Spark Streaming library provides windowed computations where the transformations on RDDs are applied over a sliding window of data.

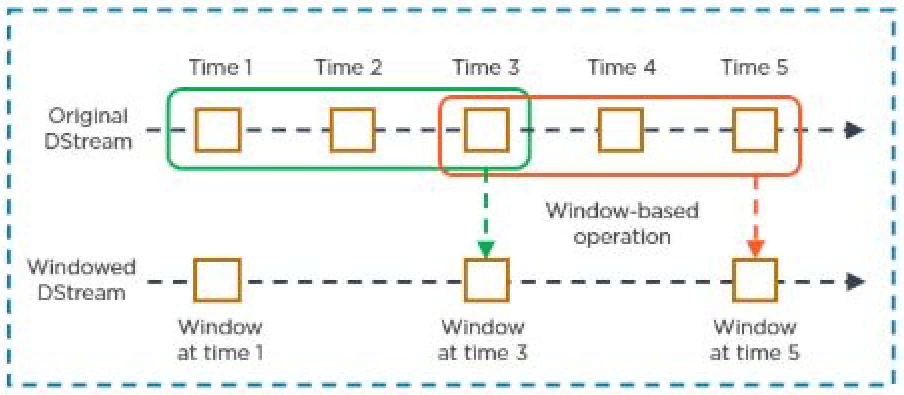


Figure 15: https://www.simplilearn.com/ice9/free__resources__article__thumb/originald-stream.JPG

**25. What are the different levels of persistence in Spark?**

DISK_ONLY - Stores the RDD partitions only on the disk

MEMORY_ONLY_SER - Stores the RDD as serialized Java objects with a one-byte array per partition

MEMORY_ONLY - Stores the RDD as deserialized Java objects in the JVM. If the RDD is not able to fit in the memory available, some partitions won't be cached

OFF_HEAP - Works like MEMORY_ONLY_SER but stores the data in off-heap memory

MEMORY_AND_DISK - Stores RDD as deserialized Java objects in the JVM. In case the RDD is not able to fit in the memory, additional partitions are stored on the disk

MEMORY_AND_DISK_SER - Identical to MEMORY_ONLY_SER with the exception of storing partitions not able to fit in the memory to the disk

**26. What is the difference between map and flatMap transformation in Spark Streaming?**

| map() | flatMap() |
|---|---|
| A map function returns a new DStream by passing each element of the source DStream through a function func | It is similar to the map function and applies to each element of RDD and it returns the result as a new RDD |
| Spark Map function takes one element as an input process it according to custom code (specified by the developer) and returns one element at a time | FlatMap allows returning 0, 1, or more elements from the map function. In the FlatMap operation |

**27. How would you compute the total count of unique words in Spark?**

1. Load the text file as RDD:

sc.textFile("hdfs://Hadoop/user/test_file.txt");

2. Function that breaks each line into words:

def toWords(line):

return line.split();

3. Run the toWords function on each element of RDD in Spark as flatMap transformation:

```
words = line.flatMap(toWords);
```

4. Convert each word into (key,value) pair:

```
def toTuple(word):
return (word, 1);
wordTuple = words.map(toTuple);
```

5. Perform reduceByKey() action:

```
def sum(x, y):
return x+y:
counts = wordsTuple.reduceByKey(sum)
```

6. Print:

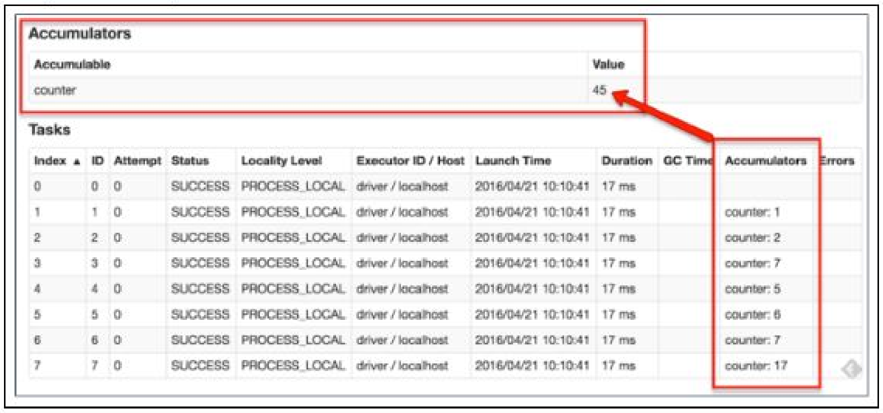
```
counts.collect()
```

### 28. Suppose you have a huge text file. How will you check if a particular keyword exists using Spark?

```
lines = sc.textFile("hdfs://Hadoop/user/test_file.txt");
def isFound(line):
if line.find("my_keyword") > -1
return 1
return 0
foundBits = lines.map(isFound);
sum = foundBits.reduce(sum);
if sum > 0:
print "Found"
else:
print "Not Found";
```

### 29. What is the role of accumulators in Spark?

Accumulators are variables used for aggregating information across the executors. This information can be about the data or API diagnosis like how many records are corrupted or how many times a library API was called.



**Accumulators**

| Accumulable | Value |
|---|---|
| counter | 45 |

**Tasks**

| Index ▲ | ID | Attempt | Status | Locality Level | Executor ID / Host | Launch Time | Duration | GC Tim | Accumulators | Errors |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | SUCCESS | PROCESS_LOCAL | driver / localhost | 2016/04/21 10:10:41 | 17 ms | | | |
| 1 | 1 | 0 | SUCCESS | PROCESS_LOCAL | driver / localhost | 2016/04/21 10:10:41 | 17 ms | | counter: 1 | |
| 2 | 2 | 0 | SUCCESS | PROCESS_LOCAL | driver / localhost | 2016/04/21 10:10:41 | 17 ms | | counter: 2 | |
| 3 | 3 | 0 | SUCCESS | PROCESS_LOCAL | driver / localhost | 2016/04/21 10:10:41 | 17 ms | | counter: 7 | |
| 4 | 4 | 0 | SUCCESS | PROCESS_LOCAL | driver / localhost | 2016/04/21 10:10:41 | 17 ms | | counter: 5 | |
| 5 | 5 | 0 | SUCCESS | PROCESS_LOCAL | driver / localhost | 2016/04/21 10:10:41 | 17 ms | | counter: 6 | |
| 6 | 6 | 0 | SUCCESS | PROCESS_LOCAL | driver / localhost | 2016/04/21 10:10:41 | 17 ms | | counter: 7 | |
| 7 | 7 | 0 | SUCCESS | PROCESS_LOCAL | driver / localhost | 2016/04/21 10:10:41 | 17 ms | | counter: 17 | |

Figure 16: https://www.simplilearn.com/ice9/free_resources_article_thumb/api.JPG

### 30. What are the different MLlib tools available in Spark?

- ML Algorithms: Classification, Regression, Clustering, and Collaborative filtering
- Featurization: Feature extraction, Transformation, Dimensionality reduction,

and Selection

- Pipelines: Tools for constructing, evaluating, and tuning ML pipelines
- Persistence: Saving and loading algorithms, models, and pipelines
- Utilities: Linear algebra, statistics, data handling

### 31. What are the different data types supported by Spark MLlib?

Spark MLlib supports local vectors and matrices stored on a single machine, as well as distributed matrices.

Local Vector: MLlib supports two types of local vectors - dense and sparse

Example: vector(1.0, 0.0, 3.0)

dense format: [1.0, 0.0, 3.0]

sparse format: (3, [0, 2]. [1.0, 3.0])

Labeled point: A labeled point is a local vector, either dense or sparse that is associated with a label/response.

Example: In binary classification, a label should be either 0 (negative) or 1 (positive)

Local Matrix: A local matrix has integer type row and column indices, and double type values that are stored in a single machine.

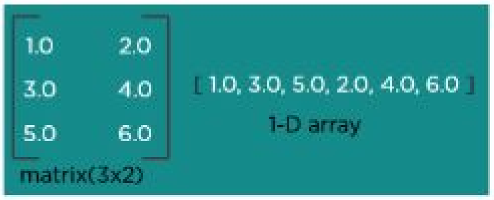


Figure 17: https://www.simplilearn.com/ice9/free_resources_article_thumb/local-matrix.JPG

Distributed Matrix: A distributed matrix has long-type row and column indices and double-type values, and is stored in a distributed manner in one or more RDDs.

Types of the distributed matrix:

- RowMatrix
- IndexedRowMatrix
- CoordinatedMatrix

### 32. What is a Sparse Vector?

A Sparse vector is a type of local vector which is represented by an index array and a value array.

public class SparseVector

extends Object

implements Vector

Example: sparse1 = SparseVector(4, [1, 3], [3.0, 4.0])

where:

4 is the size of the vector

[1,3] are the ordered indices of the vector

[3,4] are the value

Do you have a better example for this spark interview question? If yes, let us know.

### 33. Describe how model creation works with MLlib and how the model is applied.

MLlib has 2 components:

Transformer: A transformer reads a DataFrame and returns a new DataFrame with a specific transformation applied.

Estimator: An estimator is a machine learning algorithm that takes a DataFrame to train a model and returns the model as a transformer.

Spark MLlib lets you combine multiple transformations into a pipeline to apply complex data transformations.

The following image shows such a pipeline for training a model:

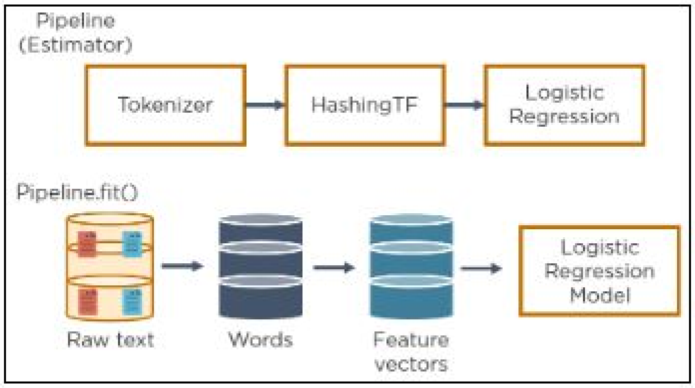


Figure 18: https://www.simplilearn.com/ice9/free_resources_article_thumb/pipeline.JPG

The model produced can then be applied to live data:

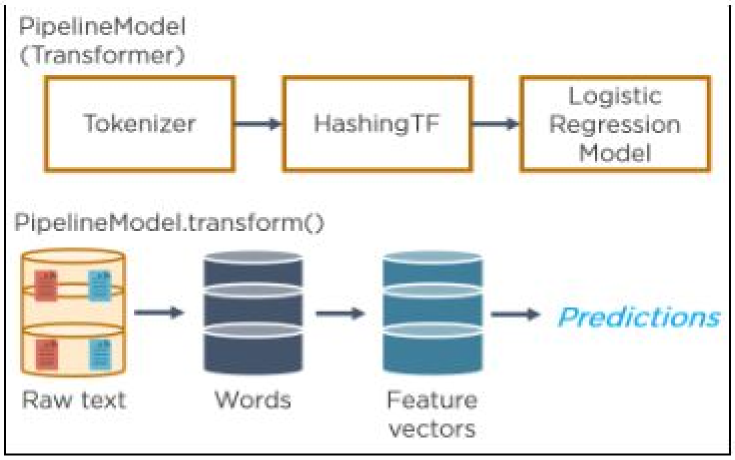


Figure 19: https://www.simplilearn.com/ice9/free_resources_article_thumb/live-data.JPG

**34. What are the functions of Spark SQL?**

Spark SQL is Apache Spark's module for working with structured data.

Spark SQL loads the data from a variety of structured data sources.

It queries data using SQL statements, both inside a Spark program and from external tools that connect to Spark SQL through standard database connectors (JDBC/ODBC).

It provides a rich integration between SQL and regular Python/Java/Scala code, including the ability to join RDDs and SQL tables and expose custom functions in SQL.

**35. How can you connect Hive to Spark SQL?**

To connect Hive to Spark SQL, place the hive-site.xml file in the conf directory of Spark.

XML
hive-site.xml
conf directory

Figure 20: https://www.simplilearn.com/ice9/free_resources_article_thumb/hive-spark.JPG

Using the Spark Session object, you can construct a DataFrame.

result=spark.sql("select * from ")

**36. What is the role of Catalyst Optimizer in Spark SQL?**

Catalyst optimizer leverages advanced programming language features (such as Scala's pattern matching and quasi quotes) in a novel way to build an extensible query optimizer.

**37. How can you manipulate structured data using domain-specific language in Spark SQL?**

Structured data can be manipulated using domain-Specific language as follows:

Suppose there is a DataFrame with the following information:

val df = spark.read.json("examples/src/main/resources/people.json")

// Displays the content of the DataFrame to stdout

df.show()

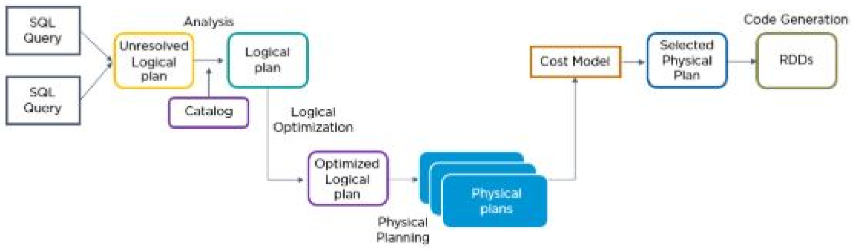


Figure 21: https://www.simplilearn.com/ice9/free_resources_article_thumb/catalyst-optimizer.JPG

```
// +—-+——-+
// | age| name|
// +—-+——-+
// |null|Michael|
// | 30| Andy|
// | 19| Justin|
// +—-+——-+
// Select only the “name” column
df.select(“name”).show()
// +——-+
// | name|
// +——-+
// |Michael|
// | Andy|
// | Justin|
// +——-+
// Select everybody, but increment the age by 1
df.select($“name”, $“age” + 1).show()
// +——-+———+
// | name|(age + 1)|
// +——-+———+
// |Michael| null|
// | Andy| 31|
// | Justin| 20|
// +——-+———+
// Select people older than 21
df.filter($“age” > 21).show()
// +—+—-+
// |age|name|
// +—+—-+
// | 30|Andy|
// +—+—-+
// Count people by age
df.groupBy(“age”).count().show()
// +—-+—–+
// | age|count|
// +—-+—–+
// | 19| 1|
// |null| 1|
// | 30| 1|
// +—-+—–+
```

**38. What are the different types of operators provided by the Apache GraphX library?**

In such spark interview questions, try giving an explanation too (not just the name of the operators).

Property Operator: Property operators modify the vertex or edge properties using a user-defined map function and produce a new graph.

Structural Operator: Structure operators operate on the structure of an input graph and produce a new graph.

Join Operator: Join operators add data to graphs and generate new graphs.

**39. What are the analytic algorithms provided in Apache Spark GraphX?**

GraphX is Apache Spark's API for graphs and graph-parallel computation. GraphX includes a set of graph algorithms to simplify analytics tasks. The algorithms are contained in the org.apache.spark.graphx.lib package and can be accessed directly as methods on Graph via GraphOps.

PageRank: PageRank is a graph parallel computation that measures the importance of each vertex in a graph. Example: You can run PageRank to evaluate what the most important pages in Wikipedia are.

Connected Components: The connected components algorithm labels each connected component of the graph with the ID of its lowest-numbered vertex. For example, in a social network, connected components can approximate clusters.

Triangle Counting: A vertex is part of a triangle when it has two adjacent vertices with an edge between them. GraphX implements a triangle counting algorithm in the TriangleCount object that determines the number of triangles passing through each vertex, providing a measure of clustering.

**40. What is the PageRank algorithm in Apache Spark GraphX?**

It is a plus point if you are able to explain this spark interview question thoroughly, along with an example! PageRank measures the importance of each vertex in a graph, assuming an edge from u to v represents an endorsement of v's importance by u.

u
v's importance by u
v

Figure 22: https://www.simplilearn.com/ice9/free_resources_article_thumb/u-v.JPG

If a Twitter user is followed by many other users, that handle will be ranked high.

PageRank algorithm was originally developed by Larry Page and Sergey Brin to rank websites for Google. It can be applied to measure the influence of vertices in any network graph. PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The assumption is that more important websites are likely to receive more links from other websites.

Figure 23: https://www.simplilearn.com/ice9/free_resources_article_thumb/twitter.JPG

### 41. What's Spark Driver?

Spark Driver is the programme that runs on the machine's master node and tells RDDs how to be changed and what to do with them. In simple terms, a Spark driver creates a SparkContext linked to a specific Spark Master.

The driver also sends the RDD graphs to Master, where the cluster manager runs independently.

### 42. What is the Spark Executor?

SparkContext gets an Executor on each node in the cluster when it connects to a cluster manager. Executors are Spark processes that run computations and store the results on the worker node. SparkContext gives the last tasks to executors so that they can be done.

### 43. What do you mean when you say "worker node"?

A worker node is any node in a cluster that can run the application code. The driver programme must listen for connections from its executors and accept them. It must also be reached on the network from the worker nodes.

The worker node is the slave node. The master node gives out work, and the worker nodes do the job. The data stored on the node is processed by the worker nodes, which then report the resources to the master. The master schedules tasks based on whether or not there are enough resources.

### 44. What's a sparse vector?

A sparse vector has two parallel arrays, one for the indices and the other for the values. To save space, these vectors store entries that are not zero.

### 45. Can data stored in Cassandra databases be accessed and analysed using Spark?

If you use Spark Cassandra Connector, you can do it.

A Cassandra Connector will need to be added to the Spark project to connect Spark to a Cassandra cluster. During setup, a Spark executor will talk to a local Cassandra node and only ask for locally stored data. It speeds up queries by sending data between Spark executors (which process data) and Cassandra nodes with less network use (where data lives).

### 46. Can Apache Spark be used with Apache Mesos?

Yes, Mesos-managed hardware clusters can run Apache Spark. In the diagram below, the cluster manager is a Spark master instance used when a cluster is set up independently. When Mesos is used, the Mesos master takes over as the cluster manager from the Spark master. Mesos decides what tasks each machine will do. To avoid the need for static resource partitioning, it considers other frameworks when scheduling these numerous temporary tasks.

### 47. What are the variables that are broadcast?

With broadcast variables, a programmer can keep a cached copy of a read-only variable on each machine instead of sending a copy with each task. They can be used to quickly give each node its copy of a large input dataset. Spark also tries to spread out variables that are broadcast using efficient broadcast algorithms to lower the cost of communication.

### 48. Tell me about Apache Spark's accumulators.

Accumulators are variables that can only be added with an operation that works both ways. They are used to do things like count or add. Keeping track of accumulators in the UI can help you understand how running stages are going. Spark supports numeric accumulators by default. We can make accumulators with or without names.

### 49. Why are broadcast variables important when working with Apache Spark?

Broadcast variables can only be read, and every machine has them in its memory cache. Using broadcast variables when working with Spark, you don't have to send copies of a variable for each task. This lets data be processed faster. Broadcast variables make it possible to store a lookup table in memory, which makes retrieval faster than with an RDD lookup ().

### 50. How can you set up automatic cleanups in Spark to deal with metadata that has built up?

You can start the cleanups by splitting long-running jobs into batches and writing the intermediate results to disc.

### 51. What does it mean to operate in a sliding window fashion?

Sliding Window controls how data packets move from one computer network to another. Spark Streaming offers windowed computations, in which changes to RDDs are made over a sliding data window. When the window moves, the RDDs that fall within the new window are added together and processed to make new RDDs of the windowed DStream.

### 52. Tell me about Spark Streaming's caching.

Developers can store or cache the stream's data in memory with DStreams. This is helpful if the DStream data will be computed more than once. A DStream's persist() method can be used to do this. For input streams that get data over the network (like Kafka, Flume, Sockets, etc. ), the default persistence level is set to copy the data to two nodes so that if one goes down, the other one will still have the data.

### 53. Is it necessary to install Spark on all the nodes of a YARN cluster when running Spark applications?

Spark can run jobs on YARN and Mesos without needing installation, as it can operate atop these clusters with little to no modification required.

### 54. Does Apache Spark have checkpoints?

Checkpoints work like checkpoints in video games. They ensure it works 24/7 and can handle failures that have nothing to do with the application logic.

The use of lineage graphs is essential for restoring RDDs after a failure, but if the RDDs have lengthy lineage chains, this process can be time-consuming. Spark has an API for checkpointing, the same as a REPLICATE flag for keeping data. But it is up to the user to decide which data to check. When lineage graphs are extended and have many connections, checkpoints are helpful.

### 55. What does Spark do with Akka?

Akka is mainly used by Spark for scheduling. After signing up, every worker asks for a task to learn. The master gives the task. Spark uses Akka to facilitate communication between the workers and the masters in this scenario.

### 56. What does "lazy evaluation" mean to you?

Spark is brilliant in how it works with data. When you tell Spark to work on a particular dataset, it listens to your instructions and writes them down so it doesn't forget, but it doesn't do anything until you ask for the result. When a function like a map() is called on an RDD, the change doesn't happen immediately. In Spark, transformations aren't evaluated until you do something. This helps improve the way data is processed as a whole.

### 57. In Apache Spark RDD, what does SchemaRDD mean?

SchemaRDD is an RDD made up of row objects, which are just wrappers for basic arrays of strings or integers, and schema information about the data type in each column.

SchemaRDD made it easier for developers to debug code and do unit tests on the SparkSQL core module in their daily work. The idea can be summed up by saying that the data structures inside RDD should be described formally, like a relational database schema. SchemaRDD gives you some simple relational query interface functions that you can use with SparkSQL on top of the essential functions that most RDD APIs offer.

### 58. What's different about Spark SQL from SQL and HQL?

Spark SQL is a particular part of the Spark Core engine that works with Hive Query Language and SQL without changing the syntax. Spark SQL can combine SQL tables and HQL tables.

### 59. Give an example of when you use Spark Streaming.

Regarding Spark Streaming, the data flows into our Spark programme in real-time.

Spark Streaming is used in the real world to analyse how people feel about things on Twitter. Trending topics can be used to make campaigns that reach more people. It helps with managing crises, making changes to services, and marketing to specific groups.

The sentiment is how someone feels about something they say on social media. Sentiment analysis is putting tweets about a specific topic into groups and using Sentiment Automation Analytics Tools to mine data.

With Spark Streaming, the Spark programme can get live tweets from all over the world. We can use Spark SQL to filter this stream, and then we can filter tweets based on how they make us feel. The logic for filtering will be built with MLlib, which lets us learn from how people think and change our filtering scale to match.

### 60. What does RDD mean?

Resilient Distributed Datasets is the name of Spark's primary abstraction. Resilient Distributed Datasets are pieces of data that are split up and have these qualities. The most popular RDD properties are immutable, distributed, lazy evaluation, and catchable.

**61. Please explain what can't change.**

Once a value has been made and given, it can no longer be changed. The name for this quality is immutability. Spark is always the same. It doesn't work with upgrades or changes. Please remember that the data storage is not immutable, but the information itself is.

**62. How does RDD get the word out?**

RDD can send data to different parallel computing nodes in a way that changes over time.

**63. What are Spark's different Ecosystems?**

Here are some typical Spark ecosystems:

- Spark SQL for SQL developers
- Spark Streaming for data streaming
- MLLib for machine learning algorithms
- GraphX for graph computing
- SparkR to work with the Spark engine
- BlinkDB, which lets you ask questions about large amounts of data in real-time.

**64. What are walls made of?**

Partition is a way to divide records logically. This idea comes from Map-Reduce (split), which uses logical data to process data directly. Small bits of data can also help the operation grow and go faster. Input data, output data & intermediate data are all partitioned RDDs.

**65. How does Spark divide up data?**

The map-reduce API is used for the data partition in Spark. In the input format, one can make more than one partition. For best performance, the HDFS block size is the partition size, but you can change partition sizes with tools like Split.

**66. How does Spark store data?**

A computer without a storage engine is called a spark. It can get information from any storage engine, like S3, HDFS, and other services.

**67. Do you have to run the Hadoop programme to run Spark?**

Spark has no particular storage, but you don't have to do it. So, you must store the files using the local file system. You can load data from a local device and work with it. To run a Spark programme, you do not need Hadoop or HDFS.

### 68. What is SparkContext?

When a programmer makes RDDs, SparkContext makes a new SparkContext object by connecting to the Spark cluster. SparkContext lets Spark know how to move around the cluster. SparkConf is an essential part of building an app for a programmer.

### 69. What's different about SparkSQL from HQL and SQL?

SparkSQL is a particular part of the SparkCore engine that supports SQL and HiveQueryLanguage without changing the syntax. You will enter both the SQL table and the HQL table.

### 70. When do you use Spark streaming?

A programme interface (API) streams data and processes it in real-time. Spark streaming gets streaming data from services like web server log files, social media data, stock market data, and Hadoop ecosystems like Kafka and Flume.

### 71. How do you use the Spark Streaming API?

In the setup, the programmer must choose a certain amount of time during which the data that goes into Spark is split into batches. The stream that comes in (called "DStream") goes into the Spark stream.

The framework breaks up into small pieces called batches, which are then sent to the Spark engine to be processed. The batches are sent to the central engine by the Spark Streaming API. The final results from core engines can be streamed in batches. Even the way things are made is in batches. It lets data be processed both as it comes in and all at once.

### 72. What does GraphX mean?

GraphX is an API for Spark that lets you change graphics and arrays. It brings together ETL, analysis, and iterative graph computing. It has the fastest graphics system, which can handle mistakes and is easy to use without special training.

### 74. What does File System API stand for?

The File System API can read data from HDFS, S3, and Local FileSystem, among other storage devices. Spark uses the FS API to get information from different storage engines.

### 75. Why are partitions immutable?

With each change, a new partition is made. Partitions use the HDFS API, so they can't be changed, are spread out, and can handle mistakes. Because of this, partitions are aware of where the results are.

**76. Explain what Spark's flatMap and Map are.**

A map is a simple line or row used to process data. FlatMap can map each input object to several different output items. So it is usually used to produce the Array's parts.

**77. Explain what broadcast variables are.**

With broadcast variables, a programmer can send a copy of a read-only variable with each task. Instead, the variable is cached on each computer. Spark has two types of mutual variables: broadcast variables and accumulators. Broadcast variables are kept in Array Buffers, which send values that can only be read to the nodes that are doing work.

**78. About Hadoop, what are Spark Accumulators?**

Accumulators are the name for Spark debuggers that work offline. Spark accumulators are like Hadoop counters in that they can track how many activities are going on. The accumulator's value can only be read by the driver programme, not the tasks.

**79. When can you use Apache Spark? What is better about Spark than MapReduce?**

Spark moves pretty quickly. Hadoop MapReduce is ten times slower in memory than other programming frameworks. It uses RAM in the right way so that it works faster.

In Map Reduce Paradigm, you write a lot of Map-Reduce tasks and then use the Oozie/shell script to link these tasks together. This process takes a long time, and the role of map-reducing is slow.

Changing production from one MR job to another MR job can sometimes require writing more code because Oozie may need to be more.

Spark lets you do everything from a single application or console and get the results immediately. It's pretty easy to switch between "doing something locally" and "Running something on a cluster." This means that the creator has less background change and can work faster.

**80. Is MapReduce learning good for anything?**

Yes. It is used for the following:

- Spark and other big data tools use MapReduce, a way of doing things. So it's essential to learn the MapReduce model and how to turn a problem into a series of MR tasks.
- The Hadoop Map-Reduce model is critical when data grows beyond what can fit in the cluster memory.

- Almost every other tool, like Hive or Pig, changes the query into a series of MapReduce steps. If you understand MapReduce, you'll be able to make better queries.

**81. What does RDD Lineage mean?**

Spark doesn't let you copy data in memory, so if you lose data, you must rebuild it using RDD lineage. It is a process that puts together data partitions that have been lost. RDD always remembers how to build from other datasets, which is the best thing about it.

**82. What does Spark not do well?**

Spark remembers things. This is something that the developer needs to be careful with. Developers who are not careful can make the following mistakes:

- It might run everything on the local node instead of sending work to the cluster.
- By using multiple clusters, it could call some web services too many times. The Hadoop MapReduce model is an excellent way to solve the first problem.
- Map-Reduce can also go wrong in a second way. When using Map-Reduce, users can touch the service too often from within the map() or reduce() functions. This is also likely to happen when using Spark.