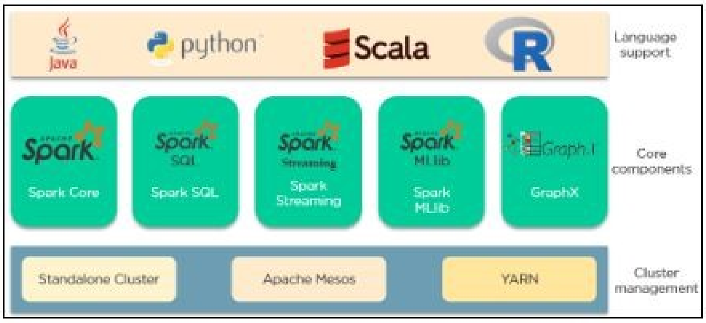
Apache spark interview questions and answers

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 longer time to retrieve the data
Spark provides caching and in-memory data storage	Hadoop is highly disk-dependent

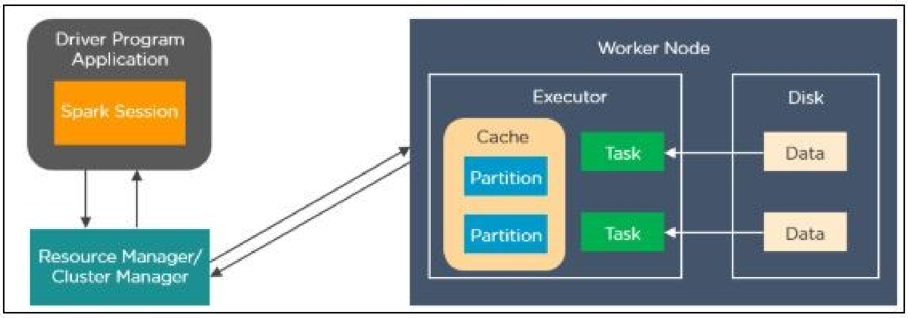
2. What are the important components of the Spark ecosystem?



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.



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?

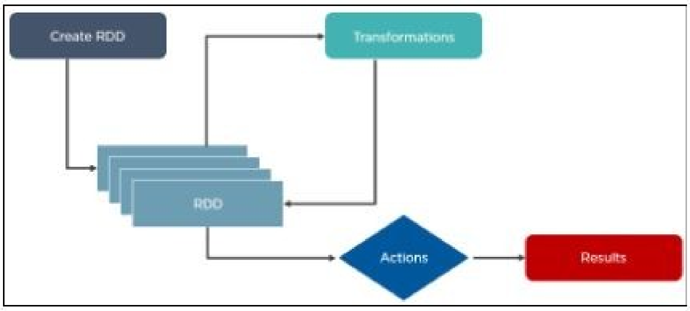
- **Standalone Mode**: By default, applications submitted to the standalone mode cluster will run in FIFO order, and each application will try to use all available nodes. You can launch a standalone cluster either manually, by starting a master and workers by hand or use our provided launch scripts. 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 is 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:



6. What is a lazy evaluation in Spark?

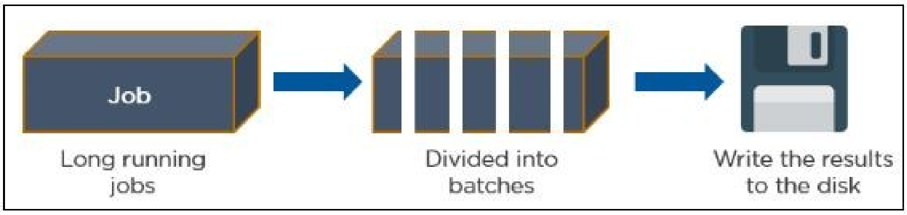
When Spark operates on any dataset, it remembers the instructions.When a transformation such as a map() is called on an RDD, the operation is not performed instantly. Transformations in Spark are not evaluated until you perform an action, which aids in optimizing the overall data processing workflow,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



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:

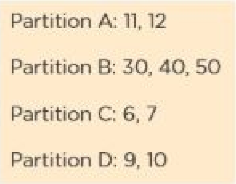
spark.shuffle.compress – checks whether the engine would compress shuffle outputs or not
spark.shuffle.spill.compress – decides whether to compress intermediate shuffle spill files or not

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.



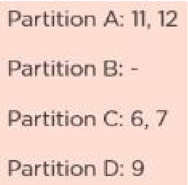
Partition A: 11, 12

Partition B: 30, 40, 50

Partition C: 6, 7

Partition D: 9, 10

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



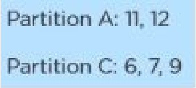
Partition A: 11, 12

Partition B: -

Partition C: 6, 7

Partition D: 9

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



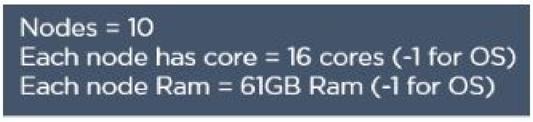
Partition A: 11, 12

Partition C: 6, 7, 9

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

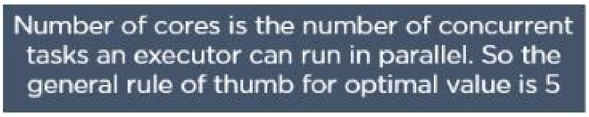
13. How can you calculate the executor memory?

Consider the following cluster information:



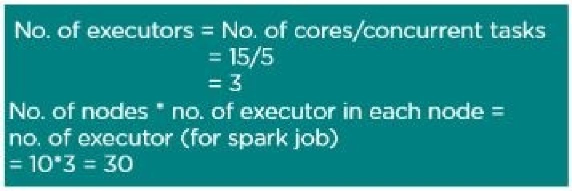
Nodes = 10
Each node has core = 16 cores (-1 for OS)
Each node Ram = 61GB Ram (-1 for OS)

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

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
```

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(<table-name>)

.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.

RDDs support 2 types of operation:

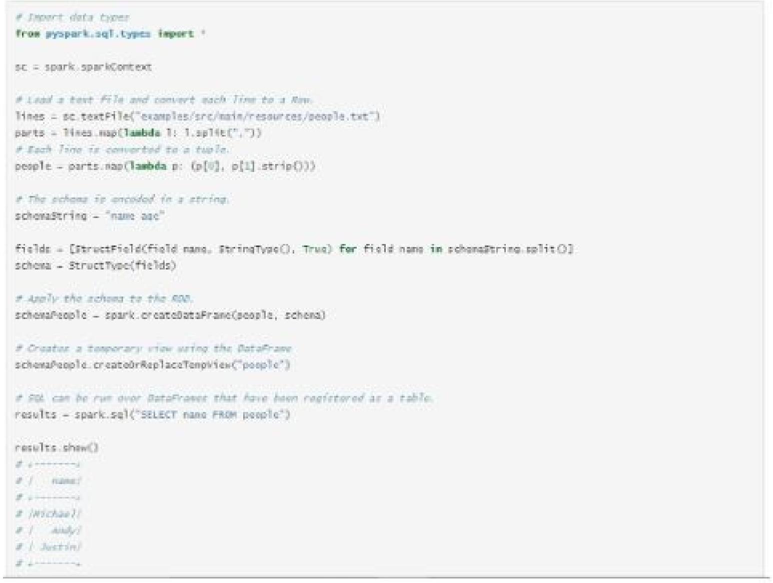
Transformations: Transformations are operations that are performed on an RDD to create a new RDD containing the results (Example: map, filter, join, union)

Actions: Actions are operations that return a value after running a computation on an RDD (Example: reduce, first, count)

17. 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|
# +-------+
```

18. What is a Lineage Graph?

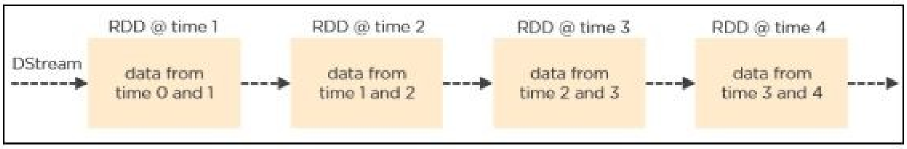
A Lineage Graph is a dependencies graph between the existing RDD and the new RDD. It means that all the dependencies between the RDD will be recorded in a graph, rather than the original data.

The need for an RDD lineage graph happens when we want to compute 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.

Q19. What do you understand about DStreams in Spark?

Discretized Streams is the basic abstraction provided by Spark Streaming.

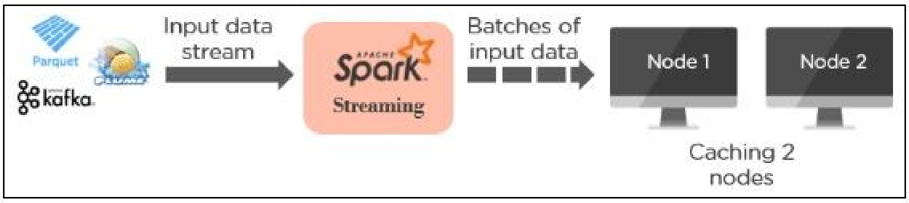
It represents a continuous stream of data that is either in the form of an input source or processed data stream generated by transforming the input stream.



20. Explain Caching in Spark Streaming.

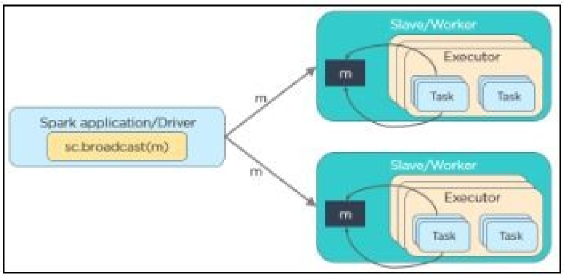
Caching also known as Persistence are optimization techniques for Spark computations. Similar to RDDs, DStreams also allow developers to persist the stream’s data in memory. That is, using the **persist()** method on a DStream will automatically persist every RDD of that DStream in memory. It helps to 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.



21. 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.



```
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)
```

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?

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.

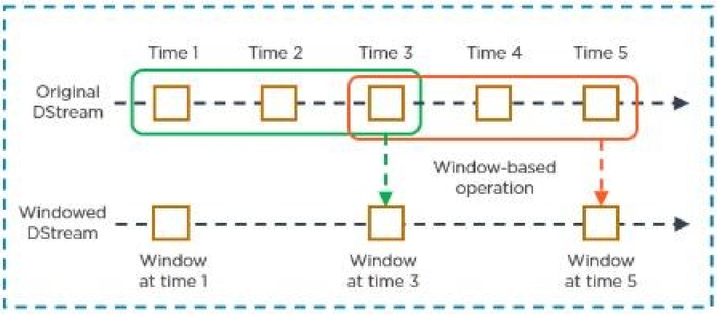
There are 2 types of data for which we can use checkpointing in Spark.

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.



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 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 map function and applies to each element of RDD and it returns the result as new RDD
Spark Map function takes one element as 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()
```

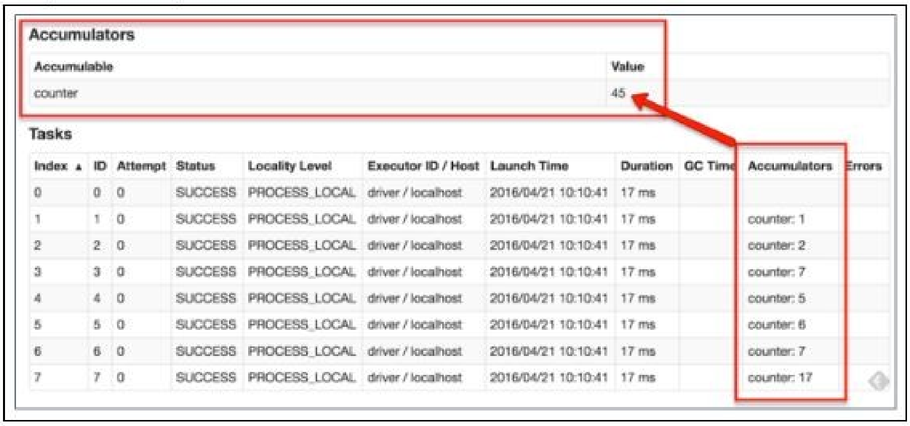
28. Suppose you have a huge text file. How will you check if a particular keyword exists using Spark?

```
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 Time	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	

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)

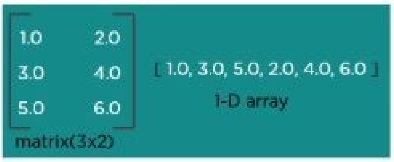
dense format: [1.0, 0.0, 3.0]

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.



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 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

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

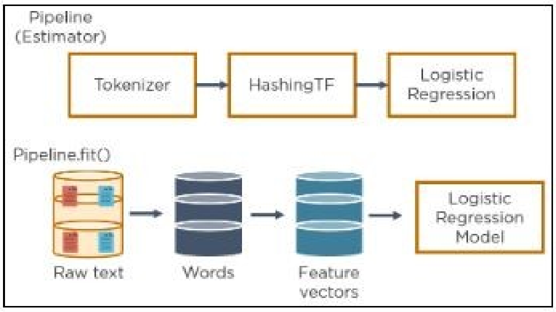
MLlib has 2 components:

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

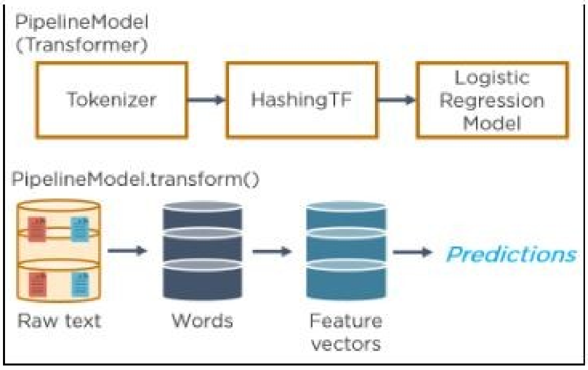
Estimator: An estimator is a machine learning algorithm that takes a DataFrame to train a model and returns the model as a transformer.

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

The following image shows such pipeline for training a model:



The model produced can then be applied to live data:



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.

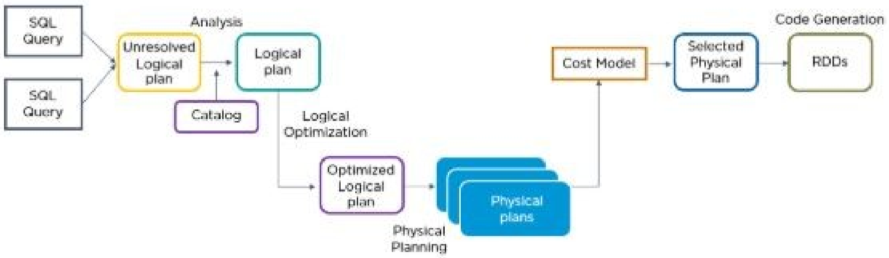
XML
hive-site.xml
conf directory

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

result=spark.sql("select * from <hive_table>")

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()

// +----+-------+

// | 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?

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**.

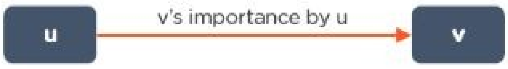
PageRank: PageRank is a graph parallel computation that measures the importance of each vertex in a graph. Example: You can run PageRank to evaluate what the most important pages in Wikipedia are.

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?

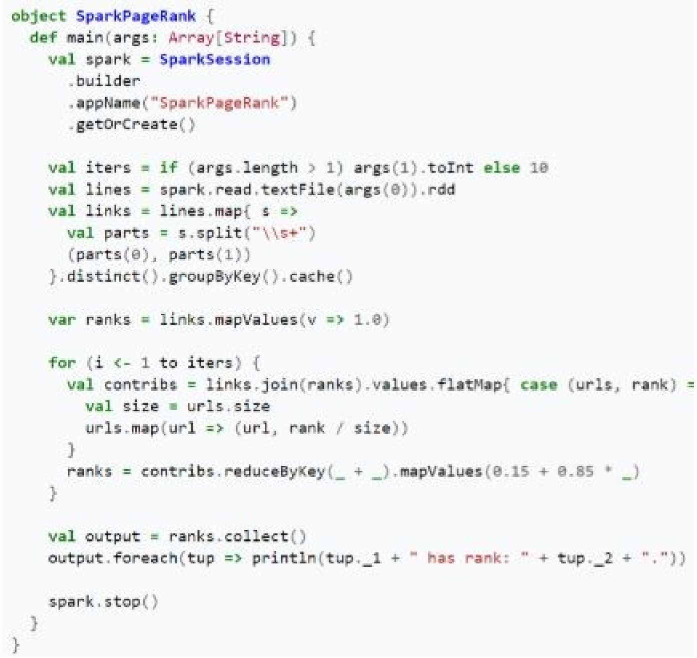
PageRank measures the importance of each vertex in a graph, assuming an edge from u to v represents an endorsement of v’s importance by u.



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.

A typical example of using Scala's functional programming with Apache Spark RDDs to iteratively compute Page Ranks is shown below:



```
object SparkPageRank {
  def main(args: Array[String]) {
    val spark = SparkSession
      .builder
      .appName("SparkPageRank")
      .getOrCreate()

    val iters = if (args.length > 1) args(1).toInt else 10
    val lines = spark.read.textFile(args(0)).rdd
    val links = lines.map{ s =>
      val parts = s.split("\\s+")
      (parts(0), parts(1))
    }.distinct().groupByKey().cache()

    var ranks = links.mapValues(v => 1.0)

    for (i <- 1 to iters) {
      val contribs = links.join(ranks).values.flatMap{ case (urls, rank) =
        val size = urls.size
        urls.map(url => (url, rank / size))
      }
      ranks = contribs.reduceByKey(_ + _).mapValues(0.15 + 0.85 * _)
    }

    val output = ranks.collect()
    output.foreach(tup => println(tup._1 + " has rank: " + tup._2 + "."))

    spark.stop()
  }
}
```

OR

1. Compare MapReduce with Spark.

Criteria	MapReduce	Spark
Processing speed	Good	Excellent (up to 100 times faster)
Data caching	Hard disk	In-memory
Performing iterative jobs	Average	Excellent
Dependency on Hadoop	Yes	No
Machine Learning applications	Average	Excellent

2. What is Apache Spark?

Spark is a fast, easy-to-use, and flexible data processing framework. It has an advanced execution engine supporting a cyclic data flow and in-memory computing. Apache Spark can run standalone, on Hadoop, or in the cloud and is capable of accessing diverse data sources including HDFS, HBase, and Cassandra, among others.

3. Explain the key features of Spark.

- Apache Spark allows integrating with Hadoop.
- It has an interactive language shell, Scala (the language in which Spark is written).
- Spark consists of RDDs (Resilient Distributed Datasets), which can be cached across the computing nodes in a cluster.
- Apache Spark supports multiple analytic tools that are used for interactive query analysis, real-time analysis, and graph processing

Learn more key features of Apache Spark in this Apache Spark Tutorial!

4. Define RDD.

RDD is the acronym for Resilient Distribution Datasets—a fault-tolerant collection of operational elements that run in parallel. The partitioned data in an RDD is immutable and distributed. There are primarily two types of RDDs:

- **Parallelized collections:** The existing RDDs running in parallel with one another
- **Hadoop datasets:** Those performing a function on each file record in HDFS or any other storage system

5. What does a Spark Engine do?

A Spark engine is responsible for scheduling, distributing, and monitoring the data application across the cluster.

Read on Spark Engine and more in this Apache Spark Community!

6. Define Partitions.

As the name suggests, a partition is a smaller and logical division of data similar to a 'split' in MapReduce. Partitioning is the process of deriving logical units of data to speed up data processing. Everything in Spark is a partitioned RDD.

7. What operations does an RDD support?

- Transformations
- Actions

8. What do you understand by Transformations in Spark?

Transformations are functions applied to RDDs, resulting in another RDD. It does not execute until an action occurs. Functions such as map() and filer() are examples of transformations, where the map() function iterates over every line in the RDD and splits into a new RDD. The filter() function creates a new RDD by selecting elements from the current RDD that passes the function argument.

9. Define Actions in Spark.

In Spark, an action helps in bringing back data from an RDD to the local machine. They are RDD operations giving non-RDD values. The reduce() function is an action that is implemented again and again until only one value if left. The take() action takes all the values from an RDD to the local node.

10. Define the functions of Spark Core.

Serving as the base engine, Spark Core performs various important functions like memory management, monitoring jobs, providing fault-tolerance, job scheduling, and interaction with storage systems.

11. What is RDD Lineage?

Spark does not support data replication in memory and thus, if any data is lost, it is rebuild using RDD lineage. RDD lineage is a process that reconstructs lost data partitions. The best thing about this is that RDDs always remember how to build from other datasets.

12. What is Spark Driver?

Spark driver is the program that runs on the master node of a machine and declares transformations and actions on data RDDs. In simple terms, a driver in Spark creates SparkContext, connected to a given Spark Master. It also delivers RDD graphs to Master, where the standalone Cluster Manager runs.

Are you interested in a comprehensive Apache Spark Training to take your career to the next level?

13. What is Hive on Spark?

Hive contains significant support for Apache Spark, wherein Hive execution is configured to Spark:

```
hive> set spark.home=/location/to/sparkHome;
hive> set hive.execution.engine=spark;
```

Hive supports Spark on YARN mode by default.

14. Name the commonly used Spark Ecosystems.

- Spark SQL (Shark) for developers
- Spark Streaming for processing live data streams
- GraphX for generating and computing graphs
- MLlib (Machine Learning Algorithms)
- SparkR to promote R programming in the Spark engine

15. Define Spark Streaming.

Spark supports stream processing—an extension to the Spark API allowing stream processing of live data streams. Data from different sources like Kafka, Flume, Kinesis is processed and then pushed to file systems, live dashboards, and databases. It is similar to batch processing in terms of the input data which is here divided into streams like batches in batch processing.

Learn in detail about the Top Four Apache Spark Use Cases including Spark Streaming!

16. What is GraphX?

Spark uses GraphX for graph processing to build and transform interactive graphs. The GraphX component enables programmers to reason about structured data at scale.

17. What does MLlib do?

MLlib is a scalable Machine Learning library provided by Spark. It aims at making Machine Learning easy and scalable with common learning algorithms and use cases like clustering, regression filtering, dimensional reduction, and the like.

18. What is Spark SQL?

Spark SQL, better known as Shark, is a novel module introduced in Spark to perform structured data processing. Through this module, Spark executes relational SQL queries on data. The core of this component supports an altogether different RDD called SchemaRDD, composed of row objects and schema objects defining the data type of each column in a row. It is similar to a table in relational databases.

Learn more about Spark from this Spark Training in New York to get ahead in your career!

19. What is a Parquet file?

Parquet is a columnar format file supported by many other data processing systems. Spark SQL performs both read and write operations with the Parquet file and considers it be one of the best Big Data Analytics formats so far.

20. What file systems does Apache Spark support?

- Hadoop Distributed File System (HDFS)
- Local file system
- Amazon S3

21. What is YARN?

Similar to Hadoop, YARN is one of the key features in Spark, providing a central and resource management platform to deliver scalable operations across the cluster. Running Spark on YARN needs a binary distribution of Spark that is built on YARN support.

Enroll in Intellipaat's Spark Course in London today to get a clear understanding of Spark!

22. List the functions of Spark SQL.

Spark SQL is capable of:

- Loading data from a variety of structured sources
- Querying data using SQL statements, both inside a Spark program and from external tools that connect to Spark SQL through standard database connectors (JDBC/ODBC), e.g., using Business Intelligence tools like Tableau
- Providing rich integration between SQL and the regular Python/Java/Scala code, including the ability to join RDDs and SQL tables, expose custom functions in SQL, and more.

23. What are the benefits of Spark over MapReduce?

- Due to the availability of in-memory processing, Spark implements data processing 10–100x faster than Hadoop MapReduce. MapReduce, on the other hand, makes use of persistence storage for any of the data processing tasks.
- Unlike Hadoop, Spark provides in-built libraries to perform multiple tasks using batch processing, steaming, Machine Learning, and interactive SQL queries. However, Hadoop only supports batch processing.
- Hadoop is highly disk-dependent, whereas Spark promotes caching and in-memory data storage.

- Spark is capable of performing computations multiple times on the same dataset, which is called iterative computation. Whereas, there is no iterative computing implemented by Hadoop.

For more insights, read on Spark vs MapReduce!

24. Is there any benefit of learning MapReduce?

Yes, MapReduce is a paradigm used by many Big Data tools, including Apache Spark. It becomes extremely relevant to use MapReduce when data grows bigger and bigger. Most tools like Pig and Hive convert their queries into MapReduce phases to optimize them better.

25. What is Spark Executor?

When SparkContext connects to Cluster Manager, it acquires an executor on the nodes in the cluster. Executors are Spark processes that run computations and store data on worker nodes. The final tasks by SparkContext are transferred to executors for their execution.

26. Name the types of Cluster Managers in Spark.

The Spark framework supports three major types of Cluster Managers.

- **Standalone:** A basic Cluster Manager to set up a cluster
- **Apache Mesos:** A generalized/commonly-used Cluster Manager, running Hadoop MapReduce and other applications
- **YARN:** A Cluster Manager responsible for resource management in Hadoop

27. What do you understand by a Worker node?

A worker node refers to any node that can run the application code in a cluster.

28. What is PageRank?

A unique feature and algorithm in GraphX, PageRank is the measure of each vertex in a graph. For instance, an edge from *u* to *v* represents an endorsement of *v*'s importance w.r.t. *u*. In simple terms, if a user at Instagram is followed massively, he/she will be ranked high on that platform.

Interested in learning Spark? Take up our Spark Training in Sydney now!

29. Do you need to install Spark on all the nodes of the YARN cluster while running Spark on YARN?

No, because Spark runs on top of YARN.

30. Illustrate some demerits of using Spark.

Since Spark utilizes more storage space when compared to Hadoop and MapReduce, there might arise certain problems. Developers need to be careful while running their applications on Spark. To resolve the issue, they can think of distributing the workload over multiple clusters, instead of running everything on a single node.

31. How to create an RDD?

Spark provides two methods to create an RDD:

- By parallelizing a collection in the driver program. This makes use of SparkContext's 'parallelize' method **val**

```
IntellipaatData = Array(2,4,6,8,10)
val distIntellipaatData = sc.parallelize(IntellipaatData)
```

- By loading an external dataset from external storage like HDFS, the shared file system

32. What are Spark DataFrames?

When a dataset is organized into SQL-like columns, it is known as a DataFrame. This is, in concept, equivalent to a data table in a relational database or a literal 'DataFrame' in R or Python. The only difference is the fact that Spark DataFrames are optimized for Big Data.

33. What are Spark Datasets?

Datasets are data structures in Spark (added since Spark 1.6) that provide the JVM object benefits of RDDs (the ability to manipulate data with lambda functions), alongside a Spark SQL-optimized execution engine.

34. Which languages can Spark be integrated with?

Spark can be integrated with the following languages:

- Python, using the Spark Python API
- R, using the R on Spark API
- Java, using the Spark Java API
- Scala, using the Spark Scala API

35. What do you mean by in-memory processing?

In-memory processing refers to the instant access of data from physical memory whenever the operation is called for. This methodology significantly reduces the delay caused by the transfer of data. Spark uses this method to access large chunks of data for querying or processing.

36. What is lazy evaluation?

Spark implements a functionality, wherein if you create an RDD out of an existing RDD or a data source, the materialization of the RDD will not occur until the RDD needs to be interacted with. This is to ensure the avoidance of unnecessary memory and CPU usage that occurs due to certain mistakes, especially in the case of Big Data Analytics.