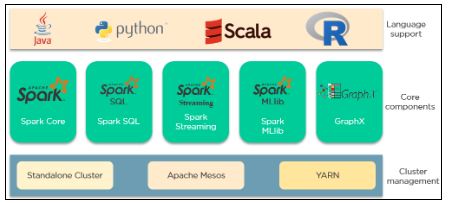
#### 1. How is Apache Spark different from MapReduce?

|  |  |
| --- | --- |
| **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](https://www.simplilearn.com/tutorials/hadoop-tutorial/mapreduce-example" \o "MapReduce data is stored in HDFS" \t "https://www.simplilearn.com/_blank) 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?

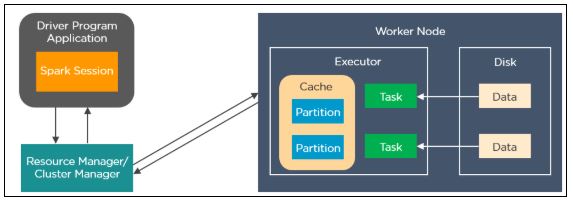


Apache Spark has 3 main categories that comprise its [ecosystem](https://www.simplilearn.com/tutorials/hadoop-tutorial/hadoop-ecosystem" \o "ecosystem" \t "https://www.simplilearn.com/_blank). Those are:

* **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.



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?

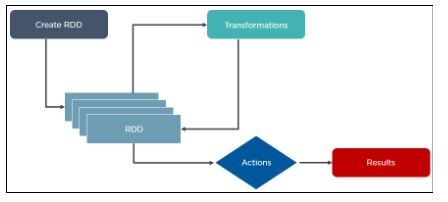
* **Standalone Mode**: By default, applications submitted to the standalone mode cluster will run in FIFO order, and each application will try to use all available nodes. You can launch a standalone cluster either manually, by starting a master and workers by hand, or use our provided launch scripts. 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](https://www.simplilearn.com/tutorials/hadoop-tutorial/yarn" \o "YARN" \t "https://www.simplilearn.com/_blank) is the cluster resource manager of Hadoop 2. Spark can be run on YARN as well.
* **Kubernetes**: [Kubernetes](https://www.simplilearn.com/tutorials/kubernetes-tutorial/what-is-kubernetes" \o "Kubernetes" \t "https://www.simplilearn.com/_blank) is an open-source system for automating deployment, scaling, and management of[containerized applications.](https://www.simplilearn.com/docker-alternatives-article" \o "containerized applications." \t "https://www.simplilearn.com/_blank)

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

Resilient Distributed Datasets are the [fundamental data structure](https://www.simplilearn.com/tutorials/data-structure-tutorial/what-is-data-structure" \o "fundamental data structure" \t "https://www.simplilearn.com/_blank) 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:



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.

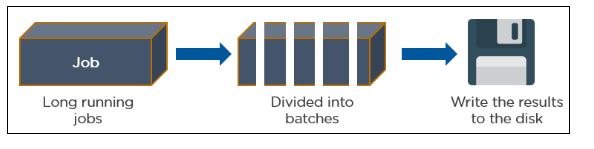
Also Read: [What Are the Skills Needed to Learn Hadoop?](https://www.simplilearn.com/learn-hadoop-article" \o "What Are the Skills Needed to Learn Hadoop?" \t "https://www.simplilearn.com/_blank)

#### 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](https://www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know-article" \o "Machine Learning algorithms" \t "https://www.simplilearn.com/_blank) 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](https://www.simplilearn.com/what-is-data-processing-article" \o "data processing" \t "https://www.simplilearn.com/_blank) 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

Learn open-source framework and scala programming languages with the [Apache Spark and Scala Certification training course](https://www.simplilearn.com/big-data-and-analytics/apache-spark-scala-certification-training?source=GhPreviewCTAText" \o "Apache Spark and Scala Certification training course" \t "https://www.simplilearn.com/_blank).

#### 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:

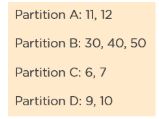
**spark.shuffle.compress** – checks whether the engine would compress shuffle outputs or not **spark.shuffle.spill.compress** – decides whether to compress intermediate shuffle spill files or not

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

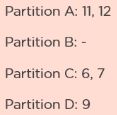
#### 12. What is the use of coalesce in Spark?

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.



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



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

IMG_263

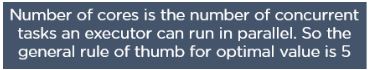
This is how the resultant RDD would look like after applying to coalesce.

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

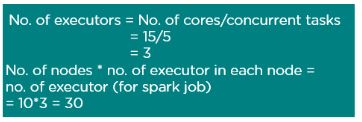
Consider the following cluster information:



Here is the number of core identification:



To calculate the number of executor identification:



#### 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.

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?

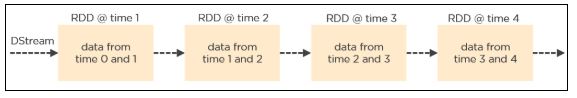
This is another frequently asked spark interview question. A Lineage Graph is a dependencies graph between the existing RDD and the new RDD. It means that all the dependencies between the RDD will be recorded in a graph,  rather than the original data.

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.

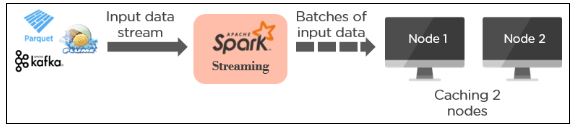
The DStreams in Spark take input from many sources such as Kafka, Flume, Kinesis, or TCP sockets. It can also work as a data stream generated by converting the input stream. It facilitates developers with a high-level API and fault tolerance.



#### 19. Explain Caching in Spark Streaming.

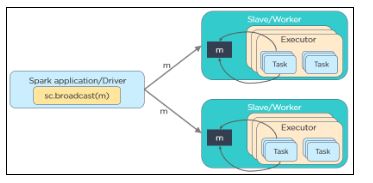
Caching also known as Persistence is an optimization technique for Spark computations. Similar to RDDs, DStreams also allow developers to persist the stream’s data in memory. That is, using the **persist()** method on a DStream will automatically persist every RDD of that DStream in memory. It helps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



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

Broadcast variables allow the [programmer](https://www.simplilearn.com/job-roles-for-programmers-article" \o "programmer" \t "https://www.simplilearn.com/_blank) 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)

So far, if you have any doubts regarding the spark interview questions for beginners, please ask in the comment section below.

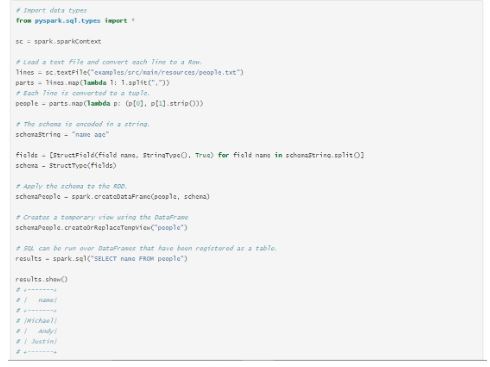
Moving forward, let us understand the spark interview questions for experienced candidates

## 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**.



#### 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.

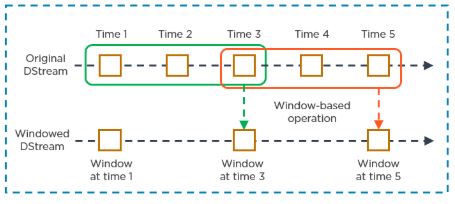
There are 2 types of data for which we can use checkpointing in Spark.

**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.



#### 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:

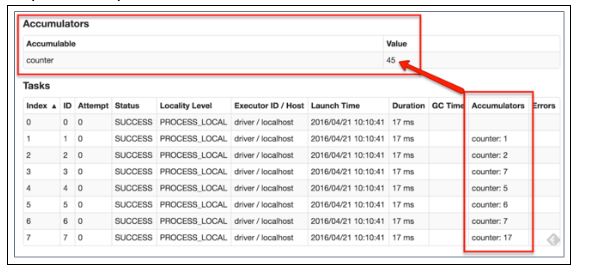
print “Found”

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.



#### 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

### ****1. Compare Hadoop and Spark.****

We will compare Hadoop MapReduce and Spark based on the following aspects:

|  |  |  |
| --- | --- | --- |
| ****Apache Spark vs. Hadoop**** ****Feature Criteria**** | ****Apache Spark**** | ****Hadoop**** |
| ****Speed**** | 100 times faster than Hadoop | Decent speed |
| ****Processing**** | Real-time & Batch processing | Batch processing only |
| ****Difficulty**** | Easy because of high level modules | Tough to learn |
| ****Recovery**** | Allows recovery of partitions | Fault-tolerant |
| ****Interactivity**** | Has interactive modes | No interactive mode except Pig & Hive |

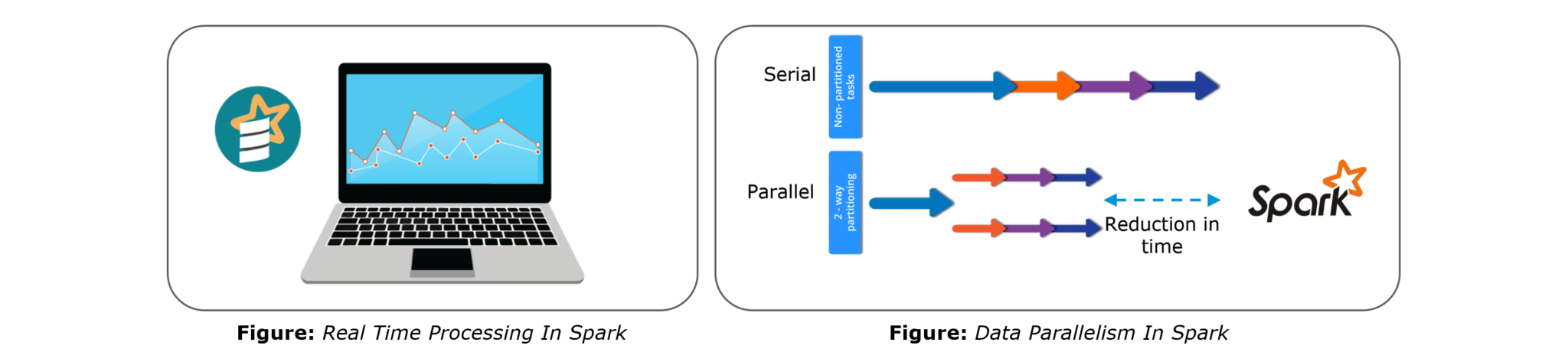
****Table:**** Apache Spark versus Hadoop

Let us understand the same using an interesting analogy.

“Single cook cooking an entree is regular computing. Hadoop is multiple cooks cooking an entree into pieces and letting each cook her piece.  
Each cook has a separate stove and a food shelf. The first cook cooks the meat, the second cook cooks the sauce. This phase is called “Map”. A the end the main cook assembles the complete entree. This is called “Reduce”. For Hadoop, the cooks are not allowed to keep things on the stove between operations. Each time you make a particular operation, the cook puts results on the shelf. This slows things down.  
For Spark, the cooks are allowed to keep things on the stove between operations. This speeds things up. Finally, for Hadoop the recipes are written in a language which is illogical and hard to understand. For Spark, the recipes are nicely written.” – Stan Kladko, Galactic Exchange.io

### ****2. What is Apache Spark?****

* **[Apache Spark](https://www.edureka.co/blog/spark-tutorial/" \t "https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/_blank)** is an open-source cluster computing framework for real-time processing.
* It has a thriving open-source community and is the most active Apache project at the moment.
* Spark provides an interface for programming entire clusters with implicit data parallelism and fault-tolerance.

Spark is of the most successful projects in the Apache Software Foundation. Spark has clearly evolved as the market leader for Big Data processing. Many organizations run Spark on clusters with thousands of nodes. Today, Spark is being adopted by major players like Amazon, eBay, and Yahoo!

### ****3. Explain the key features of Apache Spark.****

The following are the key features of Apache Spark:

1. ****Polyglot****
2. ****Speed****
3. ****Multiple Format Support****
4. ****Lazy Evaluation****
5. ****Real Time Computation****
6. ****Hadoop Integration****
7. ****Machine Learning****

Let us look at these features in detail:

****Polyglot****: Spark provides high-level APIs in Java, Scala, Python and R. Spark code can be written in any of these four languages. It provides a shell in Scala and Python. The Scala shell can be accessed through ****./bin/spark-shell**** and Python shell through ****./bin/pyspark**** from the installed directory.

****Speed****: Spark runs upto 100 times faster than Hadoop MapReduce for large-scale data processing. Spark is able to achieve this speed through controlled partitioning. It manages data using partitions that help parallelize distributed data processing with minimal network traffic.

****Multiple Formats****: Spark supports multiple data sources such as Parquet, JSON, Hive and Cassandra. The Data Sources API provides a pluggable mechanism for accessing structured data though Spark SQL. Data sources can be more than just simple pipes that convert data and pull it into Spark.

****Lazy Evaluation****: Apache Spark delays its evaluation till it is absolutely necessary. This is one of the key factors contributing to its speed. For transformations, Spark adds them to a DAG of computation and only when the driver requests some data, does this DAG actually gets executed.

****Real Time Computation****: Spark’s computation is real-time and has less latency because of its in-memory computation. Spark is designed for massive scalability and the Spark team has documented users of the system running production clusters with thousands of nodes and supports several computational models.

****Hadoop Integration****: Apache Spark provides smooth compatibility with Hadoop. This is a great boon for all the Big Data engineers who started their careers with Hadoop. Spark is a potential replacement for the MapReduce functions of Hadoop, while Spark has the ability to run on top of an existing Hadoop cluster using YARN for resource scheduling.

****Machine Learning****: Spark’s MLlib is the machine learning component which is handy when it comes to big data processing. It eradicates the need to use multiple tools, one for processing and one for machine learning. Spark provides data engineers and data scientists with a powerful, unified engine that is both fast and easy to use.

### ****4. What are the languages supported by Apache Spark and which is the most popular one?****

Apache Spark supports the following four languages: Scala, Java, Python and R. Among these languages, Scala and Python have interactive shells for Spark. The Scala shell can be accessed through ****./bin/spark-shell****and the Python shell through ****./bin/pyspark****. Scala is the most used among them because Spark is written in Scala and it is the most popularly used for Spark.

### ****5. What are benefits of Spark over MapReduce?****

Spark has the following benefits over MapReduce:

1. Due to the availability of in-memory processing, Spark implements the processing around 10 to 100 times faster than Hadoop MapReduce whereas MapReduce makes use of persistence storage for any of the data processing tasks.
2. Unlike Hadoop, Spark provides inbuilt libraries to perform multiple tasks from the same core like batch processing, Steaming, Machine learning, Interactive SQL queries. However, Hadoop only supports batch processing.
3. Hadoop is highly disk-dependent whereas Spark promotes caching and in-memory data storage.
4. Spark is capable of performing computations multiple times on the same dataset. This is called iterative computation while there is no iterative computing implemented by Hadoop.

### ****6. 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. YARN is a distributed container manager, like Mesos for example, whereas Spark is a data processing tool. Spark can run on YARN, the same way Hadoop Map Reduce can run on YARN. Running Spark on YARN necessitates a binary distribution of Spark as built on YARN support.

### ****7. Do you need to install Spark on all nodes of YARN cluster?****

No, because Spark runs on top of YARN. Spark runs independently from its installation. Spark has some options to use YARN when dispatching jobs to the cluster, rather than its own built-in manager, or Mesos. Further, there are some configurations to run YARN. They include master, deploy-mode, driver-memory, executor-memory, executor-cores, and queue.

### ****8. Is there any benefit of learning MapReduce if Spark is better than MapReduce?****

Yes, MapReduce is a paradigm used by many big data tools including Spark as well. It is extremely relevant to use MapReduce when the data grows bigger and bigger. Most tools like Pig and Hive convert their queries into MapReduce phases to optimize them better. Learn more about Big Data and its applications from the [Azure Data Engineering Certification in London](https://www.edureka.co/microsoft-azure-data-engineering-certification-course-london).

### ****9. Explain the concept of Resilient Distributed Dataset (RDD).****

RDD stands for Resilient Distribution Datasets. An RDD is a fault-tolerant collection of operational elements that run in parallel. The partitioned data in RDD is immutable and distributed in nature. There are primarily two types of RDD:

1. Parallelized Collections: Here, the existing RDDs running parallel with one another.
2. Hadoop Datasets: They perform functions on each file record in HDFS or other storage systems.

RDDs are basically parts of data that are stored in the memory distributed across many nodes. RDDs are lazily evaluated in Spark. This lazy evaluation is what contributes to Spark’s speed.

### ****10. How do we create RDDs in Spark?****

Spark provides two methods to create RDD:

1. By parallelizing a collection in your Driver program.

2. This makes use of SparkContext’s ‘parallelize’

|  |  |
| --- | --- |
| 1  2  3 | method **val** DataArray **=** Array(2,4,6,8,10)    **val** DataRDD **=** sc.parallelize(DataArray) |

3. By loading an external dataset from external storage like HDFS, HBase, shared file system.

### ****11. What is Executor Memory in a Spark application?****

Every spark application has same fixed heap size and fixed number of cores for a spark executor. The heap size is what referred to as the Spark executor memory which is controlled with the spark.executor.memory property of the **–executor-memory** flag. Every spark application will have one executor on each worker node. The executor memory is basically a measure on how much memory of the worker node will the application utilize.

### ****12. Define Partitions in Apache Spark.****

As the name suggests, partition is a smaller and logical division of data similar to ‘split’ in MapReduce. It is a logical chunk of a large distributed data set. Partitioning is the process to derive logical units of data to speed up the processing process. Spark manages data using partitions that help parallelize distributed data processing with minimal network traffic for sending data between executors. By default, Spark tries to read data into an RDD from the nodes that are close to it. Since Spark usually accesses distributed partitioned data, to optimize transformation operations it creates partitions to hold the data chunks. Everything in Spark is a partitioned RDD.

### ****13. What operations does RDD support?****

RDD (Resilient Distributed Dataset) is main logical data unit in Spark. An RDD has distributed a collection of objects. Distributed means, each RDD is divided into multiple partitions. Each of these partitions can reside in memory or stored on the disk of different machines in a cluster. RDDs are immutable (Read Only) data structure. You can’t change original RDD, but you can always transform it into different RDD with all changes you want.

RDDs support two types of operations: transformations and actions.

Transformations: Transformations create new RDD from existing RDD like map, reduceByKey and filter we just saw. Transformations are executed on demand. That means they are computed lazily.

Actions: Actions return final results of RDD computations. Actions triggers execution using lineage graph to load the data into original RDD, carry out all intermediate transformations and return final results to Driver program or write it out to file system.

### ****14. What do you understand by Transformations in Spark?****

Transformations are functions applied on RDD, resulting into another RDD. It does not execute until an action occurs. map() and filter() are examples of transformations, where the former applies the function passed to it on each element of RDD and results into another RDD. The filter() creates a new RDD by selecting elements from current RDD that pass function argument.

|  |  |
| --- | --- |
| 1  2  3 | **val** rawData**=**sc.textFile("path to/movies.txt")    **val** moviesData**=**rawData.map(x**=**>x.split("  ")) |

As we can see here, rawData RDD is transformed into moviesData RDD. Transformations are lazily evaluated.

### ****15. Define Actions in Spark.****

An action helps in bringing back the data from RDD to the local machine. An action’s execution is the result of all previously created transformations. Actions triggers execution using lineage graph to load the data into original RDD, carry out all intermediate transformations and return final results to Driver program or write it out to file system.

reduce() is an action that implements the function passed again and again until one value if left. take() action takes all the values from RDD to a local node.

|  |  |
| --- | --- |
| 1 | moviesData.saveAsTextFile(“MoviesData.txt”) |

As we can see here, moviesData RDD is saved into a text file called MoviesData.txt.

[](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/_blank)

### **[Apache Spark and Scala Certification Training Course](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/_blank)**

* *[Instructor-led Sessions](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/_blank)*
* *[Real-life Case Studies](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/_blank)*
* *[Assessments](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/_blank)*
* *[Lifetime Access](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/_blank)*

[Explore Curriculum](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/_blank)

### ****16. Define functions of SparkCore.****

Spark Core is the base engine for large-scale parallel and distributed data processing. The core is the distributed execution engine and the Java, Scala, and Python APIs offer a platform for distributed ETL application development. SparkCore performs various important functions like memory management, monitoring jobs, fault-tolerance, job scheduling and interaction with storage systems. Further, additional libraries, built atop the core allow diverse workloads for streaming, SQL, and machine learning. It is responsible for:

1. Memory management and fault recovery
2. Scheduling, distributing and monitoring jobs on a cluster
3. Interacting with storage systems

### ****17. What do you understand by Pair RDD?****

Apache defines PairRDD functions class as

|  |  |
| --- | --- |
| 1 | **class** PairRDDFunctions[K, V] **extends** Logging **with** HadoopMapReduceUtil **with** Serializable |

Special operations can be performed on RDDs in Spark using key/value pairs and such RDDs are referred to as Pair RDDs. Pair RDDs allow users to access each key in parallel. They have a reduceByKey() method that collects data based on each key and a join() method that combines different RDDs together, based on the elements having the same key.

### ****18. Name the components of Spark Ecosystem.****

1. ****Spark Core****: Base engine for large-scale parallel and distributed data processing
2. ****Spark Streaming****: Used for processing real-time streaming data
3. ****Spark SQL****: Integrates relational processing with Spark’s functional programming API
4. ****GraphX****: Graphs and graph-parallel computation
5. ****MLlib****: Performs machine learning in Apache Spark

### ****19. How is Streaming implemented in Spark? Explain with examples.****

Spark Streaming is used for processing real-time streaming data. Thus it is a useful addition to the core Spark API. It enables high-throughput and fault-tolerant stream processing of live data streams. The fundamental stream unit is DStream which is basically a series of RDDs (Resilient Distributed Datasets) to process the real-time data. The data from different sources like Flume, HDFS is streamed and finally processed to file systems, live dashboards and databases. It is similar to batch processing as the input data is divided into streams like batches.

### ****20. Is there an API for implementing graphs in Spark?****

GraphX is the Spark API for graphs and graph-parallel computation. Thus, it extends the Spark RDD with a Resilient Distributed Property Graph.

The property graph is a directed multi-graph which can have multiple edges in parallel. Every edge and vertex have user defined properties associated with it. Here, the parallel edges allow multiple relationships between the same vertices. At a high-level, GraphX extends the Spark RDD abstraction by introducing the Resilient Distributed Property Graph: a directed multigraph with properties attached to each vertex and edge.

To support graph computation, GraphX exposes a set of fundamental operators (e.g., subgraph, joinVertices, and mapReduceTriplets) as well as an optimized variant of the Pregel API. In addition, GraphX includes a growing collection of graph algorithms and builders to simplify graph analytics tasks.

### ****21. What is PageRank in 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. For example, if a Twitter user is followed by many others, the user will be ranked highly.

GraphX comes with static and dynamic implementations of PageRank as methods on the PageRank Object. Static PageRank runs for a fixed number of iterations, while dynamic PageRank runs until the ranks converge (i.e., stop changing by more than a specified tolerance). GraphOps allows calling these algorithms directly as methods on Graph.

### ****22. How is machine learning implemented in Spark?****

MLlib is 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 alike.

### ****23. Is there a module to implement SQL in Spark? How does it work?****

Spark SQL is a new module in Spark which integrates relational processing with Spark’s functional programming API. It supports querying data either via SQL or via the Hive Query Language. For those of you familiar with RDBMS, Spark SQL will be an easy transition from your earlier tools where you can extend the boundaries of traditional relational data processing.

Spark SQL integrates relational processing with Spark’s functional programming. Further, it provides support for various data sources and makes it possible to weave SQL queries with code transformations thus resulting in a very powerful tool.

The following are the four libraries of Spark SQL.

1. Data Source API
2. DataFrame API
3. Interpreter & Optimizer
4. SQL Service

### ****24. 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 Parquet file and consider it be one of the best big data analytics formats so far.

Parquet is a columnar format, supported by many data processing systems. The advantages of having a columnar storage are as follows:

1. Columnar storage limits IO operations.
2. It can fetch specific columns that you need to access.
3. Columnar storage consumes less space.
4. It gives better-summarized data and follows type-specific encoding.

### ****25. How can Apache Spark be used alongside Hadoop?****

The best part of Apache Spark is its compatibility with Hadoop. As a result, this makes for a very powerful combination of technologies. Here, we will be looking at how Spark can benefit from the best of Hadoop. Using Spark and Hadoop together helps us to leverage Spark’s processing to utilize the best of Hadoop’s HDFS and YARN.

Hadoop components can be used alongside Spark in the following ways:

1. ****HDFS****: Spark can run on top of HDFS to leverage the distributed replicated storage.
2. ****MapReduce****: Spark can be used along with MapReduce in the same Hadoop cluster or separately as a processing framework.
3. ****YARN****: Spark applications can also be run on YARN (Hadoop NextGen).
4. ****Batch & Real Time Processing****: MapReduce and Spark are used together where MapReduce is used for batch processing and Spark for real-time processing.

### ****26. What is RDD Lineage?****

Spark does not support data replication in the 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 is that RDD always remembers how to build from other datasets.

### ****27. What is Spark Driver?****

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

### ****28. What file systems does Spark support?****

The following three file systems are supported by Spark:

1. Hadoop Distributed File System (HDFS).
2. Local File system.
3. Amazon S3

### ****29. List the functions of Spark SQL.****

Spark SQL is capable of:

1. Loading data from a variety of structured sources.
2. 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). For instance, using business intelligence tools like Tableau.
3. Providing rich integration between SQL and regular Python/Java/Scala code, including the ability to join RDDs and SQL tables, expose custom functions in SQL, and more.

### ****30. What is Spark Executor?****

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

### ****31. Name types of Cluster Managers in Spark.****

The Spark framework supports three major types of Cluster Managers:

1. ****Standalone****: A basic manager to set up a cluster.
2. ****Apache Mesos****: Generalized/commonly-used cluster manager, also runs Hadoop MapReduce and other applications.
3. ****YARN****: Responsible for resource management in Hadoop.

### ****32. What do you understand by worker node?****

Worker node refers to any node that can run the application code in a cluster. The driver program must listen for and accept incoming connections from its executors and must be network addressable from the worker nodes.

Worker node is basically the slave node. Master node assigns work and worker node actually performs the assigned tasks. Worker nodes process the data stored on the node and report the resources to the master. Based on the resource availability, the master schedule tasks.

### ****33. Illustrate some demerits of using Spark.****

The following are some of the demerits of using Apache Spark:

1. Since Spark utilizes more storage space compared to Hadoop and MapReduce, there may arise certain problems.
2. Developers need to be careful while running their applications in Spark.
3. Instead of running everything on a single node, the work must be distributed over multiple clusters.
4. Spark’s “in-memory” capability can become a bottleneck when it comes to cost-efficient processing of big data.
5. Spark consumes a huge amount of data when compared to Hadoop.

### ****34. List some use cases where Spark outperforms Hadoop in processing.****

1. ****Sensor Data Processing****: Apache Spark’s “In-memory” computing works best here, as data is retrieved and combined from different sources.
2. ****Real Time Processing****: Spark is preferred over Hadoop for real-time querying of data. e.g. Stock Market Analysis, Banking, Healthcare, Telecommunications, etc.
3. ****Stream Processing****: For processing logs and detecting frauds in live streams for alerts, Apache Spark is the best solution.
4. ****Big Data Processing****:Spark runs upto 100 times faster than Hadoop when it comes to processing medium and large-sized datasets.

### ****36. Can you use Spark to access and analyze data stored in Cassandra databases?****

Yes, it is possible if you use Spark Cassandra Connector.To connect Spark to a Cassandra cluster, a Cassandra Connector will need to be added to the Spark project. In the setup, a Spark executor will talk to a local Cassandra node and will only query for local data. It makes queries faster by reducing the usage of the network to send data between Spark executors (to process data) and Cassandra nodes (where data lives).

### ****37. Is it possible to run Apache Spark on Apache Mesos?****

Yes, Apache Spark can be run on the hardware clusters managed by Mesos. In a standalone cluster deployment, the cluster manager in the below diagram is a Spark master instance. When using Mesos, the Mesos master replaces the Spark master as the cluster manager. Mesos determines what machines handle what tasks. Because it takes into account other frameworks when scheduling these many short-lived tasks, multiple frameworks can coexist on the same cluster without resorting to a static partitioning of resources.

### ****38. How can Spark be connected to Apache Mesos?****

To connect Spark with Mesos:

1. Configure the spark driver program to connect to Mesos.
2. Spark binary package should be in a location accessible by Mesos.
3. Install Apache Spark in the same location as that of Apache Mesos and configure the property ‘spark.mesos.executor.home’ to point to the location where it is installed.

### ****39. How can you minimize data transfers when working with Spark?****

Minimizing data transfers and avoiding shuffling helps write spark programs that run in a fast and reliable manner. The various ways in which data transfers can be minimized when working with Apache Spark are:

1. Using Broadcast Variable- Broadcast variable enhances the efficiency of joins between small and large RDDs.
2. Using Accumulators – Accumulators help update the values of variables in parallel while executing.

The most common way is to avoid operations ByKey, repartition or any other operations which trigger shuffles.

### ****40. What are broadcast variables?****

### 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 also attempts to distribute broadcast variables using efficient broadcast algorithms to reduce communication cost.****41. Explain accumulators in Apache Spark.****

Accumulators are variables that are only added through an associative and commutative operation. They are used to implement counters or sums. Tracking accumulators in the UI can be useful for understanding the progress of running stages. Spark natively supports numeric accumulators. We can create named or unnamed accumulators.

### ****42. Why is there a need for broadcast variables when working with Apache Spark?****

Broadcast variables are read only variables, present in-memory cache on every machine. When working with Spark, usage of broadcast variables eliminates the necessity to ship copies of a variable for every task, so data can be processed faster. Broadcast variables help in storing a lookup table inside the memory which enhances the retrieval efficiency when compared to an RDD lookup().

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

You can trigger the clean-ups by setting the parameter ‘spark.cleaner.ttl’ or by dividing the long running jobs into different batches and writing the intermediary results to the disk.

### ****44. What is the significance of Sliding Window operation?****

Sliding Window controls transmission of data packets between various computer networks. Spark Streaming library provides windowed computations where the transformations on RDDs are applied over a sliding window of data. Whenever the window slides, the RDDs that fall within the particular window are combined and operated upon to produce new RDDs of the windowed DStream.

### ****45. What is a DStream in Apache Spark?****

***Discretized Stream***(DStream) is the basic abstraction provided by Spark Streaming. It is a continuous stream of data. It is received from a data source or from a processed data stream generated by transforming the input stream. Internally, a DStream is represented by a continuous series of RDDs and each RDD contains data from a certain interval. Any operation applied on a DStream translates to operations on the underlying RDDs.

DStreams can be created from various sources like Apache Kafka, HDFS, and Apache Flume. DStreams have two operations:

1. Transformations that produce a new DStream.
2. Output operations that write data to an external system.

There are many DStream transformations possible in Spark Streaming. Let us look at **filter(*func*)**. filter(*func*) returns a new DStream by selecting only the records of the source DStream on which func returns true.

### ****46. Explain Caching in Spark Streaming.****

DStreams allow developers to cache/ persist the stream’s data in memory. This is useful if the data in the DStream will be computed multiple times. This can be done using the persist() method on a DStream. For input streams that receive data over the network (such as Kafka, Flume, Sockets, etc.), the default persistence level is set to replicate the data to two nodes for fault-tolerance.

### ****47. When running Spark applications, is it necessary to install Spark on all the nodes of YARN cluster?****

Spark need not be installed when running a job under YARN or Mesos because Spark can execute on top of YARN or Mesos clusters without affecting any change to the cluster.

### ****48. What are the various data sources available in Spark SQL?****

Parquet file, JSON datasets and Hive tables are the data sources available in Spark SQL.

### ****49. What are the various levels of persistence in Apache Spark?****

Apache Spark automatically persists the intermediary data from various shuffle operations, however, it is often suggested that users call persist () method on the RDD in case they plan to reuse it. Spark has various persistence levels to store the RDDs on disk or in memory or as a combination of both with different replication levels.

The various storage/persistence levels in Spark are:

1. MEMORY\_ONLY: Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, some partitions will not be cached and will be recomputed on the fly each time they’re needed. This is the default level.
2. MEMORY\_AND\_DISK: Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, store the partitions that don’t fit on disk, and read them from there when they’re needed.
3. MEMORY\_ONLY\_SER: Store RDD as *serialized* Java objects (one byte array per partition).
4. MEMORY\_AND\_DISK\_SER: Similar to MEMORY\_ONLY\_SER, but spill partitions that don’t fit in memory to disk instead of recomputing them on the fly each time they’re needed.
5. DISK\_ONLY: Store the RDD partitions only on disk.
6. OFF\_HEAP: Similar to MEMORY\_ONLY\_SER, but store the data in off-heap memory.

### ****50. Does Apache Spark provide checkpoints?****

Checkpoints are similar to checkpoints in gaming. They make it run 24/7 and make it resilient to failures unrelated to the application logic.

Lineage graphs are always useful to recover RDDs from a failure but this is generally time-consuming if the RDDs have long lineage chains. Spark has an API for checkpointing i.e. a REPLICATE flag to persist. However, the decision on which data to checkpoint – is decided by the user. Checkpoints are useful when the lineage graphs are long and have wide dependencies.

### ****51. How Spark uses Akka?****

Spark uses Akka basically for scheduling. All the workers request for a task to master after registering. The master just assigns the task. Here Spark uses Akka for messaging between the workers and masters.

### ****52. What do you understand by Lazy Evaluation?****

Spark is intellectual in the manner in which it operates on data. When you tell Spark to operate on a given dataset, it heeds the instructions and makes a note of it, so that it does not forget – but it does nothing, unless asked for the final result. When a transformation like map() is called on an RDD, the operation is not performed immediately. Transformations in Spark are not evaluated till you perform an action. This helps optimize the overall data processing workflow.

### ****53. What do you understand by SchemaRDD in Apache Spark RDD?****

SchemaRDD is an RDD that consists of row objects (wrappers around the basic string or integer arrays) with schema information about the type of data in each column.

SchemaRDD was designed as an attempt to make life easier for developers in their daily routines of code debugging and unit testing on SparkSQL core module. The idea can boil down to describing the data structures inside RDD using a formal description similar to the relational database schema. On top of all basic functions provided by common RDD APIs, SchemaRDD also provides some straightforward relational query interface functions that are realized through SparkSQL.

Now, it is officially renamed to DataFrame API on Spark’s latest trunk.

### ****54. How is Spark SQL different from HQL and SQL?****

Spark SQL is a special component on the Spark Core engine that supports SQL and Hive Query Language without changing any syntax. It is possible to join SQL table and HQL table to Spark SQL.

### ****55. Explain a scenario where you will be using Spark Streaming.****

When it comes to Spark Streaming, the data is streamed in real-time onto our Spark program.

Twitter Sentiment Analysis is a real-life use case of Spark Streaming. Trending Topics can be used to create campaigns and attract a larger audience. It helps in crisis management, service adjusting and target marketing.

### **1. Apache Spark Vs Hadoop?**

****Ans:****

|  |
| --- |
| ****Spark Vs Hadoop**** |

|  |  |  |
| --- | --- | --- |
| ****Features**** | ****Spark**** | ****Hadoop**** |
| Data processing | Part of Hadoop, hence batch processing | Batch Processing even for high volumes |
| Streaming Engine | Apache spark streaming - micro-batches | Map-Reduce |
| Data Flow | Direct Acyclic Graph-DAG | Map-Reduce |
| Computation Model | Collect and process | Map-Reduce batch-oriented model |
| Performance | Slow due to batch processing | Slow due to batch processing |
| Memory Management | Automatic memory management in the latest release | Dynamic and static - Configurable |
| Fault Tolerance | Recovery available without extra code | Highly fault-tolerant due to Map-Reduce |
| Scalability | Highly scalable - spark Cluster(8000 Nodes) | Highly scalable - Produces a large number of nodes |

### **2. What is Spark?**

****Ans:****Spark is a parallel data processing framework. It allows to development of fast, unified big data applications combining batch, streaming, and interactive analytics.

|  |
| --- |
| Interested in mastering Apache Spark Course? Enroll now for a FREE demo on **[Apache Spark Online Training](https://mindmajix.com/apache-spark-training" \o "Apache Spark Training" \t "https://mindmajix.com/_blank)**. |

### **3. Why Spark?**

****Ans:****Spark is the third-generation distributed data processing platform. It’s the unified big data solution for all big data processing problems such as batch, interacting, streaming processing. So it can ease many big data problems.

### **4. What is RDD?**

****Ans:****Spark’s primary core abstraction is called Resilient Distributed Datasets. RDD is a collection of partitioned data that satisfies these properties. Immutable, distributed, lazily evaluated, catchable are common RDD properties.

|  |
| --- |
| ****Do you know**[what is Apache Airflow](https://mindmajix.com/apache-airflow-tutorial" \o "Apache Airflow Tutorial" \t "https://mindmajix.com/_blank)**?**** |

### **5. What is Immutable?**

****Ans:****Once created and assign a value, it’s not possible to change, this property is called Immutability. Spark is by default immutable, it does not allow updates and modifications. Please note data collection is not immutable, but data value is immutable.

### **6. What is Distributed?**

****Ans:****RDD can automatically the data is distributed across different parallel computing nodes.

### **7. What is Lazy evaluated?**

****Ans:****If you execute a bunch of programs, it’s not mandatory to evaluate immediately. Especially in Transformations, this Laziness is a trigger.

[](https://bit.ly/3if9dmk)

### **8. What is Catchable?**

****Ans:****Keep all the data in memory for computation, rather than going to the disk. So Spark can catch the data 100 times faster than Hadoop.

### **9. What is Spark engine's responsibility?**

****Ans:****Spark is responsible for scheduling, distributing, and monitoring the application across the cluster.

### **10. What are common Spark Ecosystems?**

****Ans:****

* Spark SQL(Shark) for SQL developers,
* Spark Streaming for streaming data,
* MLLib for machine learning algorithms,
* GraphX for Graph computation,
* SparkR to run R on Spark engine,
* BlinkDB enabling interactive queries over massive data are common Spark ecosystems.  GraphX, SparkR, and BlinkDB are in the incubation stage.

|  |
| --- |
| ****Learn**[Spark vs Hadoop](https://mindmajix.com/spark/whats-better-to-learn-first-spark-vs-hadoop" \o "Spark vs Hadoop" \t "https://mindmajix.com/_blank)**What's Better to Learn First.**** |

### **11. What are Partitions?**

****Ans:****Partition is a logical division of the data, this idea is derived from Map-reduce (split). Logical data is specifically derived to process the data. Small chunks of data also it can support scalability and speed up the process. Input data, intermediate data, and output data everything is Partitioned RDD.

### **12. How does spark partition the data?**

****Ans:****Spark uses map-reduce API to do the partition the data. In Input format, we can create a number of partitions. By default, HDFS block size is partition size (for best performance), but it’s possible to change partition size like Split.

### **13. How does Spark store the data?**

****Ans:****Spark is a processing engine, there is no storage engine. It can retrieve data from any storage engine like HDFS, S3, and other data resources.

### **14. Is it mandatory to start Hadoop to run the spark application?**

****Ans:****No not mandatory, but there is no separate storage in Spark, so it uses a local file system to store the data. You can load data from the local system and process it, Hadoop or HDFS is not mandatory to run spark application.

### **15. What is SparkContext?**

****Ans:****When a programmer creates an RDDs, SparkContext connects to the Spark cluster to create a new SparkContext object. SparkContext tells spark how to access the cluster. SparkConf is a key factor to create a programmer application.

### **16. What are SparkCore functionalities?**

****Ans:****SparkCore is a base engine of the apache spark framework. Memory management, fault tolerance, scheduling, and monitoring jobs, interacting with store systems are primary functionalities of Spark.

### **17. How SparkSQL is different from HQL and SQL?**

****Ans:****SparkSQL is a special component on the spark core engine that supports SQL and HiveQueryLanguage without changing any syntax. It’s possible to join the SQL table and HQL table.

### **18. When did we use Spark Streaming?**

****Ans:****Spark Streaming is the real-time processing of [streaming data](https://mindmajix.com/spark/streaming-big-data-with-apache-spark" \o "Streaming Big Data with Apache Spark" \t "https://mindmajix.com/_blank) API. Spark streaming gathers streaming data from different resources like web server log files, social media data, stock market data, or Hadoop ecosystems like Flume, and Kafka.

### **19. How Spark Streaming API works?**

****Ans:****The programmer sets a specific time in the configuration, within this time how much data gets into the Spark, that data separates as a batch. The input stream (DStream) goes into spark streaming. The framework breaks up into small chunks called batches, then feeds into the spark engine for processing.

Spark Streaming API passes those batches to the core engine. The core engine can generate the final results in the form of streaming batches. The output also in the form of batches. It can allow streaming data and batch data for processing.

### **20. What is Spark MLlib?**

****Ans:****Mahout is a machine learning library for Hadoop, similarly, MLlib is a Spark library. MetLib provides different algorithms, that algorithms scale-out on the cluster for data processing. Most of the data scientists use this MLlib library.

|  |
| --- |
| ****Leave an Inquiry to Learn****: **[Apache Spark Course in Bangalore](https://mindmajix.com/apache-spark-training-bangalore" \o "Apache Spark Training Bangalore" \t "https://mindmajix.com/_blank)** |

### **21. What is GraphX?**

****Ans:****GraphX is a Spark API for manipulating Graphs and collections. It unifies ETL, other analysis, and iterative graph computation. It's the fastest graph system, provides fault tolerance and ease of use without special skills.

### **22. What is File System API?**

****Ans:****FS API can read data from different storage devices like HDFS, S3, or Local FileSystem. Spark uses FS API to read data from different storage engines.

### **23. Why Partitions are immutable?**

****Ans:****Every transformation generates a new partition.  Partitions use HDFS API so that partition is immutable, distributed, and fault-tolerant. Partition also aware of data locality.

### **24. What is Transformation in spark?**

****Ans:****Spark provides two special operations on RDDs called transformations and Actions. Transformation follows lazy operation and temporarily holds the data until unless called the Action. Each transformation generates/returns a new RDD. Example of transformations: Map, flatMap, groupByKey, reduceByKey, filter, co-group, join, sortByKey, Union, distinct, sample are common spark transformations.

### **25. What is Action in Spark?**

****Ans:****Actions are RDD’s operation, that value returns back to the spar driver programs, which kick off a job to execute on a cluster. Transformation’s output is an input of Actions. reduce, collect, take a sample, take, first, saveAsTextfile, saveAsSequenceFile, countByKey, for each is common actions in Apache spark.

### **26. What is RDD Lineage?**

****Ans:****Lineage is an RDD process to reconstruct lost partitions. Spark not replicate the data in memory, if data lost, RDD uses lineage to rebuild lost data. Each RDD remembers how the RDD build from other datasets.

### **27. What is Map and flatMap in Spark?**

****Ans:****The map is a specific line or row to process that data. In FlatMap each input item can be mapped to multiple output items (so the function should return a Seq rather than a single item). So most frequently used to return Array elements.

### **28. What are broadcast variables?**

****Ans:****Broadcast variables let the programmer keep a read-only variable cached on each machine, rather than shipping a copy of it with tasks. Spark supports 2 types of shared variables called broadcast variables (like Hadoop distributed cache) and accumulators (like Hadoop counters). Broadcast variables are stored as Array Buffers, which sends read-only values to work nodes.

### **29. What are Accumulators in Spark?**

****Ans:****Spark of-line debuggers are called accumulators. Spark accumulators are similar to Hadoop counters, to count the number of events and what’s happening during the job you can use accumulators. Only the driver program can read an accumulator value, not the tasks.

### **30. How RDD persist the data?**

****Ans:****There are two methods to persist the data, such as persist() to persist permanently and cache() to persist temporarily in the memory. Different storage level options there such as MEMORY\_ONLY, MEMORY\_AND\_DISK, DISK\_ONLY, and many more. Both persist() and cache() uses different options depends on the task.

### **31. When do you use apache spark? OR  What are the benefits of Spark over Mapreduce?**

****Ans:****

* Spark is really fast. As per their claims, it runs programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk. It aptly utilizes RAM to produce faster results.
* In the map-reduce paradigm, you write many Map-reduce tasks and then tie these tasks together using Oozie/shell script. This mechanism is very time-consuming and the map-reduce task has heavy latency.
* And quite often, translating the output out of one MR job into the input of another MR job might require writing another code because Oozie may not suffice.
* In Spark, you can basically do everything using a single application/console (pyspark or scala console) and get the results immediately. Switching between ‘Running something on cluster’ and ‘doing something locally’ is fairly easy and straightforward. This also leads to less context switch of the developer and more productivity.
* Spark kind of equals to MapReduce and Oozie put together.

### **32. Is there is a point of learning MapReduce, then?**

****Ans:****Yes. For the following reason:

* [MapReduce](https://en.wikipedia.org/wiki/MapReduce" \o "MapReduce" \t "https://mindmajix.com/_blank) is a paradigm used by many big data tools including Spark. So, understanding the MapReduce paradigm and how to convert a problem into series of MR tasks is very important.
* When the data grows beyond what can fit into the memory on your cluster, the Hadoop Map-Reduce paradigm is still very relevant.
* Almost, every other tool such as Hive or Pig converts its query into MapReduce phases. If you understand Mapreduce then you will be able to optimize your queries better.

### **33. When running Spark on Yarn, do I need to install Spark on all nodes of Yarn Cluster?**

****Ans:****Since spark runs on top of Yarn, it utilizes yarn for the execution of its commands over the cluster’s nodes.  
So, you just have to install Spark on one node.

### **34. What are the downsides of Spark?**

****Ans:****Spark utilizes memory. The developer has to be careful. A casual developer might make the following mistakes:

* She may end up running everything on the local node instead of distributing work over to the cluster.
* She might hit some web service too many times by the way of using multiple clusters.

The first problem is well tackled by Hadoop Map reduce paradigm as it ensures that the data your code is churning is fairly small at a point in time thus you can make the mistake of trying to handle whole data on a single node.  
The second mistake is possible in Map-Reduce too. While writing Map-Reduce, the user may hit a service from inside of map() or reduce() too many times. This overloading of service is also possible while using Spark.

### **35. What is an RDD?**

****Ans:****The full form of RDD is a resilient distributed dataset. It is a representation of data located on a network that is

* ****Immutable:**** You can operate on the RDD to produce another RDD but you can’t alter it.
* ****Partitioned / Parallel:**** The data located on RDD is operated in parallel. Any operation on RDD is done using multiple nodes.
* ****Resilience:**** If one of the nodes hosting the partition fails, another node takes its data.

RDD provides two kinds of operations: Transformations and Actions.

### **36. What are Transformations?**

****Ans:****The transformations are the functions that are applied on an RDD (resilient distributed data set). The transformation results in another RDD. A transformation is not executed until an action follows.

The example of transformations are:

1. ****map():**** applies the function passed to it on each element of RDD resulting in a new RDD.
2. ****filter():**** creates a new RDD by picking the elements from the current RDD which pass the function argument.

### **37. What are Actions?**

****Ans:****An action brings back the data from the RDD to the local machine. Execution of action results in all the previously created transformations. The example of actions are:

* ****reduce():****executes the function passed again and again until only one value is left. The function should take two arguments and return one value.
* ****take():**** take all the values back to the local node from RDD.

### **38. Say I have a huge list of numbers in RDD(say myrdd). And I wrote the following code to compute the average?**

****Ans:****

def myAvg(x, y):  
return (x+y)/2.0;  
avg = myrdd.reduce(myAvg);

### **39. What is wrong with it? And How would you correct it?**

****Ans:****The average function is not commutative and associative;  
I would simply sum it and then divide it by count.

def sum(x, y):

return x+y;

total = myrdd.reduce(sum);

avg = total / myrdd.count();

The only problem with the above code is that the total might become very big thus overflow. So, I would rather divide each number by count and then sum in the following way.

cnt = myrdd.count();

def devideByCnd(x):

return x/cnt;

myrdd1 = myrdd.map(devideByCnd);

avg = myrdd.reduce(sum);

### **40. Say I have a huge list of numbers in a file in HDFS. Each line has one number. And I want to compute the square root of the sum of squares of these numbers. How would you do it?**

****Ans:****

# We would first load the file as RDD from HDFS on a spark

numsAsText = sc.textFile(“hdfs://hadoop1.knowbigdata.com/user/student/sgiri/mynumbersfile.txt”);

# Define the function to compute the squaresdef toSqInt(str):

v = int(str);

return v\*v;

#Run the function on spark rdd as transformation

nums = numsAsText.map(toSqInt);

#Run the summation as reduce action

total = nums.reduce(sum)

#finally compute the square root. For which we need to import math.

import math;

print math.sqrt(total);

### **41. Is the following approach correct? Is the sqrtOfSumOfSq a valid reducer?**

****Ans:****

numsAsText =sc.textFile(“hdfs://hadoop1.knowbigdata.com/user/student/sgiri/mynumbersfile.txt”);

def toInt(str):

return int(str);

nums = numsAsText.map(toInt);

def sqrtOfSumOfSq(x, y):

return math.sqrt(x\*x+y\*y);

total = nums.reduce(sum)

import math;

print math.sqrt(total);

Yes. The approach is correct and sqrtOfSumOfSq is a valid reducer.

### **42. Could you compare the pros and cons of your approach (in Question 2 above) and my approach (in Question 3 above)?**

****Ans:****You are doing the square and square root as part of the reducing action while I am squaring in the map() and summing in reduce in my approach.

My approach will be faster because in your case the reducer code is heavy as it is calling math.sqrt() and reducer code is generally executed approximately n-1 times the spark RDD.

The only downside of my approach is that there is a huge chance of integer overflow because I am computing the sum of squares as part of the map.

### **43. If you have to compute the total counts of each of the unique words on a spark, how would you go about it?**

****Ans:****

#This will load the bigtextfile.txt as RDD in the sparklines =

sc.textFile(“hdfs://hadoop1.knowbigdata.com/user/student/sgiri/bigtextfile.txt”);

#define a function that can break each line into words

def toWords(line):

return line.split();

# Run the towards function on each element of RDD on spark as flatMap transformation.  
# We are going to flatMap instead of the map because our function is returning multiple values.

words = lines.flatMap(toWords);

# Convert each word into (key, value) pair. Her key will be the word itself and her value will be 1.

def toTuple(word):

return (word, 1);

wordsTuple = words.map(toTuple);

# Now we can easily do the reduceByKey() action.

def sum(x, y):

return x+y;

counts = wordsTuple.reduceByKey(sum)

# Now, print

counts.collect()

### **44. In a very huge text file, you want to just check if a particular keyword exists. How would you do this using Spark?**

****Ans:****

lines = sc.textFile(“hdfs://hadoop1.knowbigdata.com/user/student/sgiri/bigtextfile.txt”);

def isFound(line):

if line.find(“mykeyword”) > -1:

return 1;

return 0;

foundBits = lines.map(isFound);

sum = foundBits.reduce(sum);

if sum > 0:

print “FOUND”;

else:

print “NOT FOUND”;

### **45. Can you improve the performance of this code in the previous answer?**

****Ans:****Yes. The search is not stopping even after the word we are looking for has been found. Our map code would keep executing on all the nodes which are very inefficient.

We could utilize accumulators to report whether the word has been found or not and then stop the job. Something on these lines:

import thread, threading

from time import sleep

result = “Not Set”

lock = threading.Lock()

accum = sc.accumulator(0)

def map\_func(line):

#introduce delay to emulate the slowness

sleep(1);

if line.find(“Adventures”) > -1:

accum.add(1);

return 1;

return 0;

def start\_job():

global result

try:

sc.setJobGroup(“job\_to\_cancel”, “some description”)

lines = sc.textFile(“hdfs://hadoop1.knowbigdata.com/user/student/sgiri/wordcount/input/big.txt”);

result = lines.map(map\_func);

result.take(1);

except Exception as e:

result = “Cancelled”

lock.release()

def stop\_job():

while accum.value < 3 :

sleep(1);

sc.cancelJobGroup(“job\_to\_cancel”)

supress = lock.acquire()

supress = thread.start\_new\_thread(start\_job, tuple())

supress = thread.start\_new\_thread(stop\_job, tuple())

supress = lock.acquire()

[/tab]

### ****1. What is Apache Spark?****

Spark is a fast, easy-to-use, and flexible data processing framework. It is an open-source analytics engine that was developed by using [Scala](https://intellipaat.com/blog/what-is-scala/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank), [Python](https://intellipaat.com/blog/tutorial/python-tutorial/what-is-python/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank), [Java](https://intellipaat.com/blog/tutorial/java-tutorial/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank), and [R](https://intellipaat.com/blog/tutorial/r-programming/introduction/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank). It has an advanced execution engine supporting acyclic data flow and in-memory computing. It uses in-memory caching and optimized execution of queries for faster query analytics of data of any size. [Apache Spark](https://intellipaat.com/blog/what-is-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.

### ****2. Explain the key features of Spark.****

* Apache Spark allows integrating with [Hadoop](https://intellipaat.com/blog/tutorial/hadoop-tutorial/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank).
* 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
* Apache Spark supports stream processing in real-time.
* Spark helps in achieving a very high processing speed of data, which it achieves by reducing the read or write operations to disk.
* Apache Spark codes can be reused for data streaming, running ad-hoc queries, batch processing, etc.
* Spark is considered a better cost-efficient solution when compared to Hadoop.

***Learn more key features of Apache Spark in this [Apache Spark Tutorial](https://intellipaat.com/blog/tutorial/spark-tutorial/)!***

### ****3. What is MapReduce?****

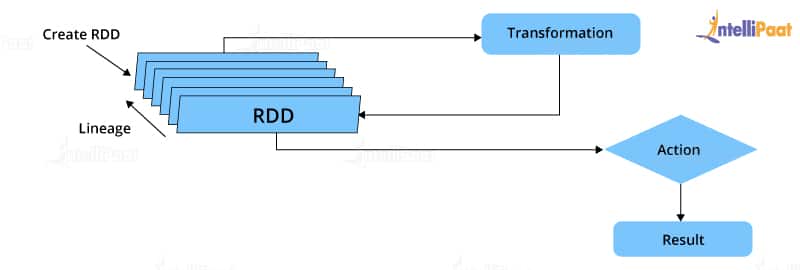
It is a software framework and programming model which is used for processing huge datasets. [MapReduce](https://intellipaat.com/blog/tutorial/mapreduce-tutorial/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank) is basically split into two parts, Map and Reduce. Map handles data splitting and data mapping, meanwhile, Reduce handles shuffle and reduction in data.

### ****4. 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 |

### ****5. Define RDD.****

RDD is the acronym for [Resilient Distribution Datasets](https://intellipaat.com/blog/tutorial/spark-tutorial/programming-with-rdds/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank)—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:

RDD in Spark

* 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

### ****6. What does a Spark Engine do?****

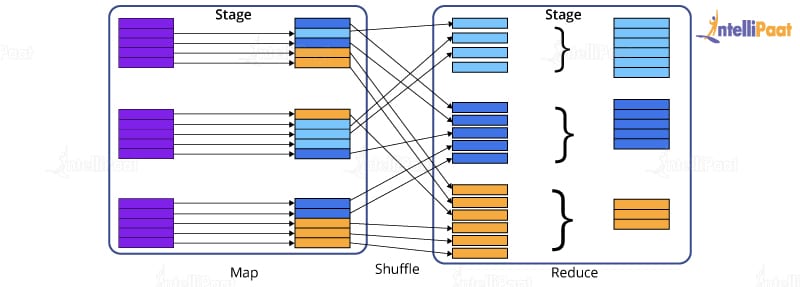
A Spark engine is responsible for scheduling, distributing, and monitoring the data application across the cluster. Spark Engine is used to run mappings in Hadoop clusters. It is suitable for wide-ranging circumstances. It includes SQL batch and ETL jobs in Spark, streaming data from sensors, IoT, ML, etc.

***Read on Spark Engine and more in this [Apache Spark Community](https://intellipaat.com/community/big-data-hadoop-spark)!***

窗体底端

### ****7. 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.



### ****8. What operations does an RDD support?****

* **Transformations:**Transformations produce a new RDD from an existing RDD, every time we apply a transformation to the RDD. Always it takes an RDD as input and ejects one or more RDD as output.
* **Actions:**Actions are used when we wish to use the actual RDD instead of working with a new RDD after we apply transformations. Actions eject out non-RDD values unlike transformations, which only eject RDD values.

### ****9. What do you understand about 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.

### ****10. Define Actions in Spark.****

Actions are operations in Spark; they help in working with the actual data set. They help in transferring data from executor to driver. In Spark, an action helps in bringing back data from an RDD to the local machine. They are RDD operations giving non-RDD values, which is unlike transformations operations, which only eject RDD as output. The reduce() function is an action that is implemented again and again until only one value is left. The take() action takes all the values from an RDD to the local node.

### ****Check out this insightful video on Spark Tutorial for Beginners:****

### ****11. Define the functions of Spark Core.****

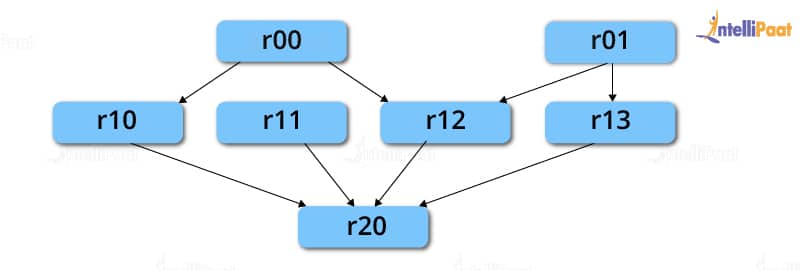
Serving as the base engine, Spark Core performs various important functions like memory management, basic I/O functionalities, monitoring jobs, providing fault-tolerance, job scheduling, interaction with storage systems, distributed task dispatching, and many more. Spark Core is the base of all projects. The above-mentioned functions are Spark Core’s primary functions.

***Learn more about Spark from this [Spark Training in New York](https://intellipaat.com/apache-spark-scala-training-new-york/) to get ahead in your career!***

## **Intermediate Interview Questions**

### ****12. What is RDD Lineage?****

Spark does not support data replication in memory and thus, if any data is lost, it is rebuilt 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.

**Career Transition**

### ****13. 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.

### ****14. What is Hive on Spark?****

[Hive](https://intellipaat.com/blog/tutorial/hadoop-tutorial/apache-hive/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank) 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;

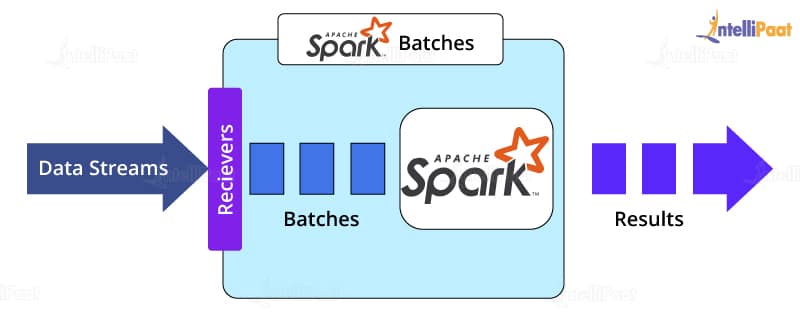
Hive supports Spark on YARN mode by default.

### ****15. Name the commonly used Spark Ecosystems.****

* [Spark SQL](https://intellipaat.com/blog/what-is-spark-sql/) (Shark) for developers
* Spark Streaming for processing live data streams
* GraphX for generating and computing graphs
* MLlib ([Machine Learning Algorithms](https://intellipaat.com/blog/tutorial/machine-learning-tutorial/machine-learning-algorithms/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank))
* SparkR to promote R Programming in the Spark engine

### ****16. 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](https://intellipaat.com/blog/top-4-apache-spark-use-cases/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank)**including Spark Streaming!****

### ****17. 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.

### ****18. What does MLlib do?****

MLlib is a scalable Machine Learning library provided by Spark. It aims at making [Machine Learning](https://intellipaat.com/blog/what-is-machine-learning/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank) easy and scalable with common learning algorithms and use cases like clustering, regression filtering, dimensional reduction, and the like.

### ****19. 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.

### ****20. 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 to be one of the best [Big Data Analytics](https://intellipaat.com/blog/big-data-analytics/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank) formats so far.

**Courses you may like**

### ****21. What file systems does Apache Spark support?****

Apache Spark is a powerful distributed data processing engine that processes data coming from multiple data sources. The file systems that Apache Spark supports are:

* [Hadoop Distributed File System (HDFS)](https://intellipaat.com/blog/tutorial/hadoop-tutorial/hdfs-overview/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank)
* Local file system
* [Amazon S3](https://intellipaat.com/blog/what-is-amazon-s3/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank)
* [HBase](https://intellipaat.com/blog/what-is-apache-hbase/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank)
* [Cassandra](https://intellipaat.com/blog/what-is-apache-cassandra/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank), etc.

### ****22. What is Directed Acyclic Graph in Spark?****

Directed Acyclic Graph or DAG is an arrangement of edges and vertices. As the name implies the graph is not cyclic. In this graph, the vertices represent RDDs, and the edges represent the operations applied to RDDs. This graph is unidirectional, which means it has only one flow. DAG is a scheduling layer that implements stage-oriented scheduling and converts a plan for logical execution to a physical execution plan.

### ****23.What are deploy modes in Apache Spark?****

There are only two deploy modes in Apache Spark, client mode and cluster mode. The behavior of Apache Spark jobs depends on the driver component. If the driver component of Apache Spark will run on the machine from which the job is submitted, then it is the client mode. If the driver component of Apache Spark will run on Spark clusters and not on the local machine from which the job is submitted, then it is the cluster mode.

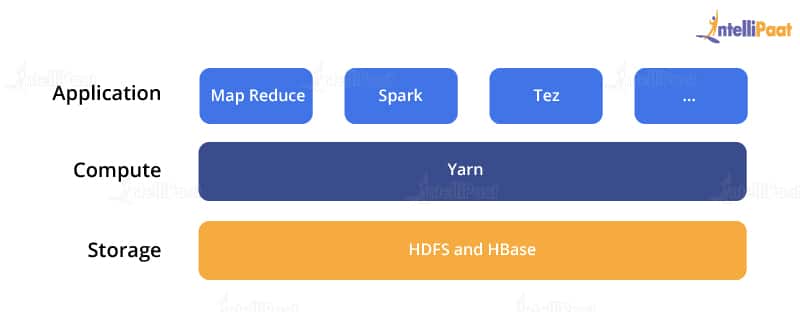
### ****24. Roles of receivers in Apache Spark Streaming?****

Within [Apache Spark Streaming](https://intellipaat.com/blog/a-guide-to-apache-spark-streaming-tutorial/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank) Receivers are special objects whose only goal is to consume data from different data sources and then move it to Spark. You can create receiver objects by streaming contexts as long-running tasks on various executors. There are two types of receivers. They are ****Reliable receivers:****This receiver acknowledges data sources when data is received and replicated successfully in Apache Spark Storage. ****Unreliable receiver:****These receivers do not acknowledge data sources even when they receive or replicate in Apache Spark Storage.

## **Advanced Spark Interview Questions**

### ****25. What is YARN?****

Similar to Hadoop, [YARN](https://intellipaat.com/blog/tutorial/hadoop-tutorial/what-is-yarn/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank) 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](https://intellipaat.com/apache-spark-scala-training-london/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank) today to get a clear understanding of Spark!**

### ****26. 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.

### ****27. 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](https://intellipaat.com/blog/spark-vs-map-reduce/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank)**!****

### ****28. 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.

### ****29. What is a 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.

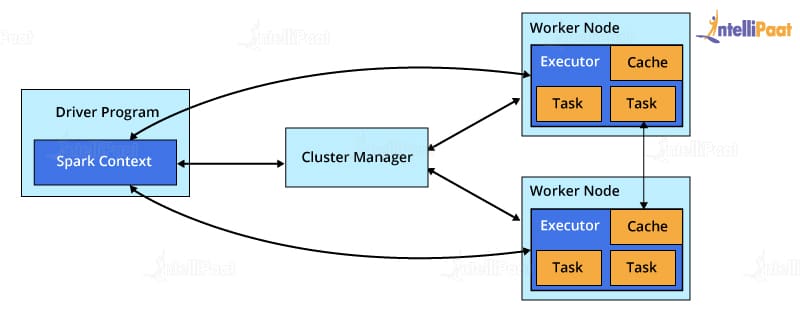
### ****30. Name the types of Cluster Managers in Spark.****

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

### ****31. What do you understand by a Worker node?****

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



### ****32. 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 on Instagram is followed massively, he/she will be ranked high on that platform.

### ****33. 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.

### ****34. 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 of Spark](https://intellipaat.com/blog/tutorial/spark-tutorial/apache-spark-applications/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank). To resolve the issue, they can think of distributing the workload over multiple clusters, instead of running everything on a single node.

### ****35. 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****

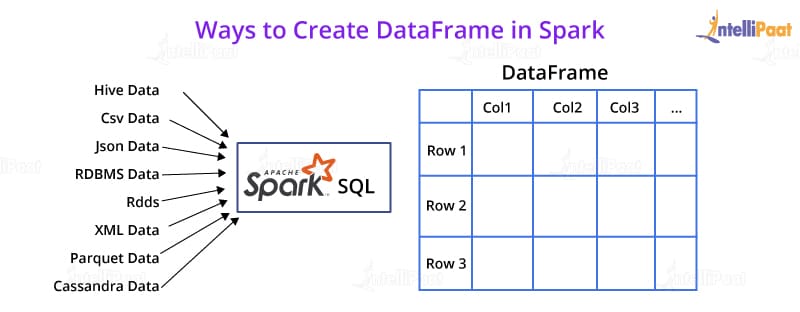
IntellipaatData = Array(2,4,6,8,10)

val distIntellipaatData = sc.parallelize(IntellipaatData)

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

### ****36. 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](https://intellipaat.com/blog/tutorial/spark-tutorial/spark-dataframe/" \t "https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/_blank) are optimized for Big Data.

### ****37. 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.

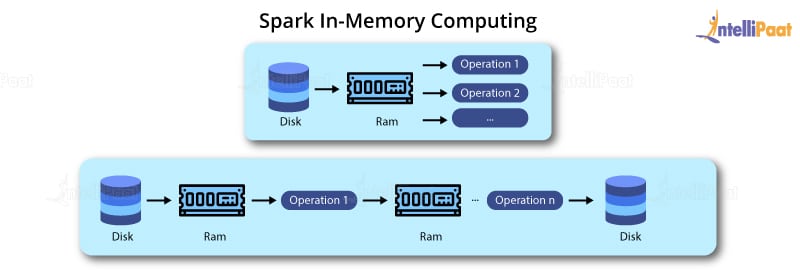
### ****38. Which languages can Spark be integrated with?****

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

### ****39. 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.

### ****40. 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.