**Spark Interview Questions**

[**Apache Spark**](https://www.interviewbit.com/blog/apache-spark-architecture/) is an open-source, lightning-fast computation technology built based on Hadoop and MapReduce technologies that support various computational techniques for fast and efficient processing. Spark is known for its in-memory cluster computation which is the main contributing feature for increasing the processing speed of the spark applications. Spark was developed as part of Hadoop’s subproject by Matei Zaharia in 2009 at UC Berkeley’s AMPLab. It was later open-sourced in the year 2010 under the BSD License which was then donated to the Apache Software Foundation in the year 2013. From 2014 onwards, Spark grabbed the top-level position among all the projects undertaken by Apache Foundation.

This article covers the most commonly asked interview questions in Spark technology and helps the [software engineers](https://www.interviewbit.com/blog/software-engineer/) and the [data engineers](https://www.interviewbit.com/blog/data-engineer/) equip themselves for the interview. The questions range from basic to intermediate to advanced levels based on the Spark concepts.

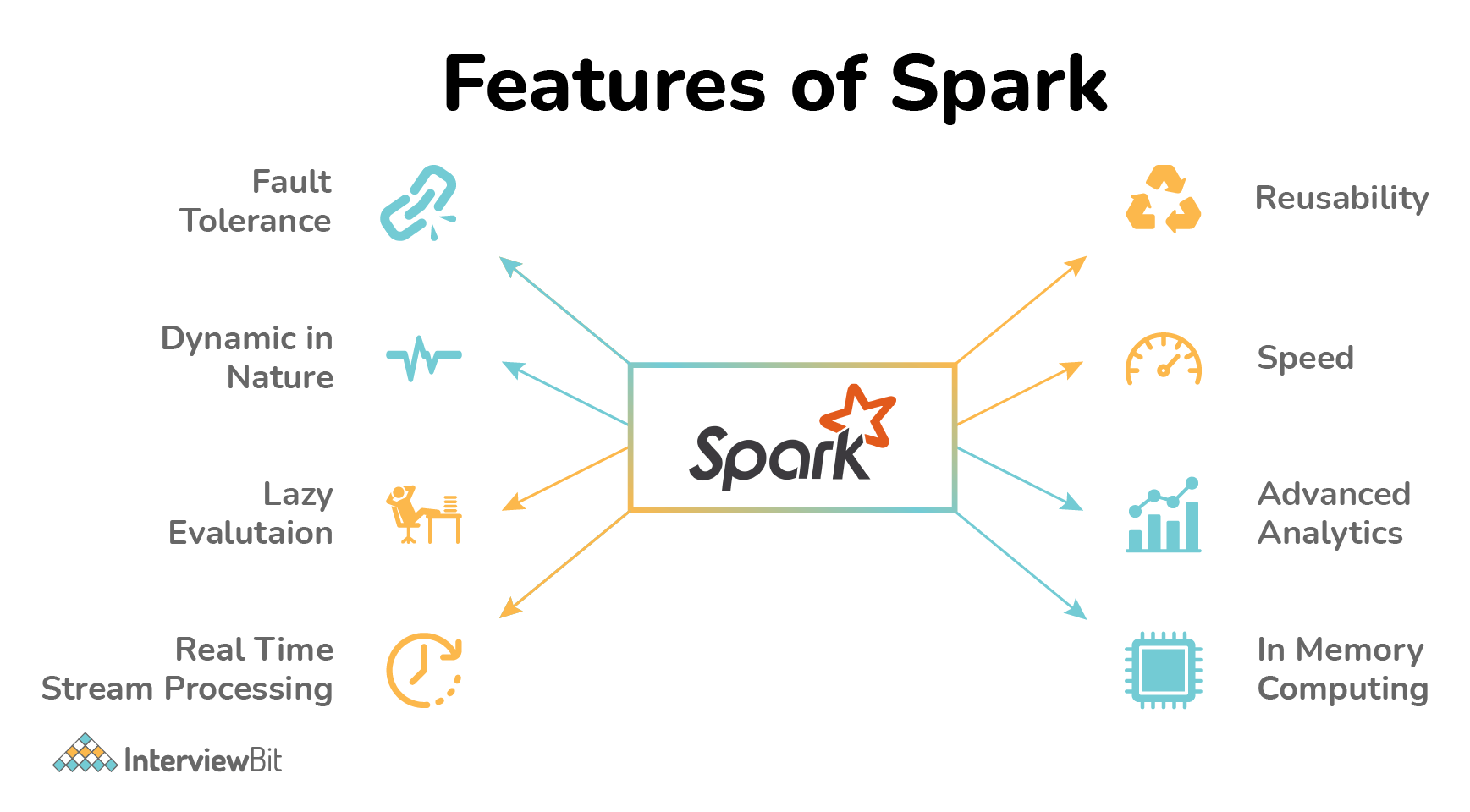
Live Masterclass

## Spark Interview Questions for Freshers

### 1. What is Apache Spark?

Apache Spark is an open-source framework engine that is known for its speed, easy-to-use nature in the field of big data processing and analysis. It also has built-in modules for graph processing, machine learning, streaming, SQL, etc. The spark execution engine supports in-memory computation and cyclic data flow and it can run either on cluster mode or standalone mode and can access diverse data sources like HBase, HDFS, Cassandra, etc.

### 2. What are the features of Apache Spark?



* **High Processing Speed**: Apache Spark helps in the achievement of a very high processing speed of data by reducing read-write operations to disk. The speed is almost 100x faster while performing in-memory computation and 10x faster while performing disk computation.
* **Dynamic Nature**: Spark provides 80 high-level operators which help in the easy development of parallel applications.
* **In-Memory Computation**: The in-memory computation feature of Spark due to its DAG execution engine increases the speed of data processing. This also supports data caching and reduces the time required to fetch data from the disk.
* **Reusability**: Spark codes can be reused for batch-processing, data streaming, running ad-hoc queries, etc.
* **Fault Tolerance**: Spark supports fault tolerance using RDD. Spark RDDs are the abstractions designed to handle failures of worker nodes which ensures zero data loss.
* **Stream Processing**: Spark supports stream processing in real-time. The problem in the earlier MapReduce framework was that it could process only already existing data.
* **Lazy Evaluation**: Spark transformations done using Spark RDDs are lazy. Meaning, they do not generate results right away, but they create new RDDs from existing RDD. This lazy evaluation increases the system efficiency.
* **Support Multiple Languages**: Spark supports multiple languages like R, Scala, Python, Java which provides dynamicity and helps in overcoming the Hadoop limitation of application development only using Java.
* **Hadoop Integration**: Spark also supports the Hadoop YARN cluster manager thereby making it flexible.
* **Supports Spark GraphX** for graph parallel execution, Spark SQL, libraries for Machine learning, etc.
* **Cost Efficiency**: Apache Spark is considered a better cost-efficient solution when compared to Hadoop as Hadoop required large storage and data centers while data processing and replication.
* **Active Developer’s Community**: Apache Spark has a large developers base involved in continuous development. It is considered to be the most important project undertaken by the Apache community.

### 3. What is RDD?

RDD stands for Resilient Distribution Datasets. It is a fault-tolerant collection of parallel running operational elements. The partitioned data of RDD is distributed and immutable. There are two types of datasets:

* **Parallelized collections**: Meant for running parallelly.
* **Hadoop datasets**: These perform operations on file record systems on HDFS or other storage systems.

**You can download a PDF version of Spark Interview Questions.**

[**Download PDF**](javascript:void(0))

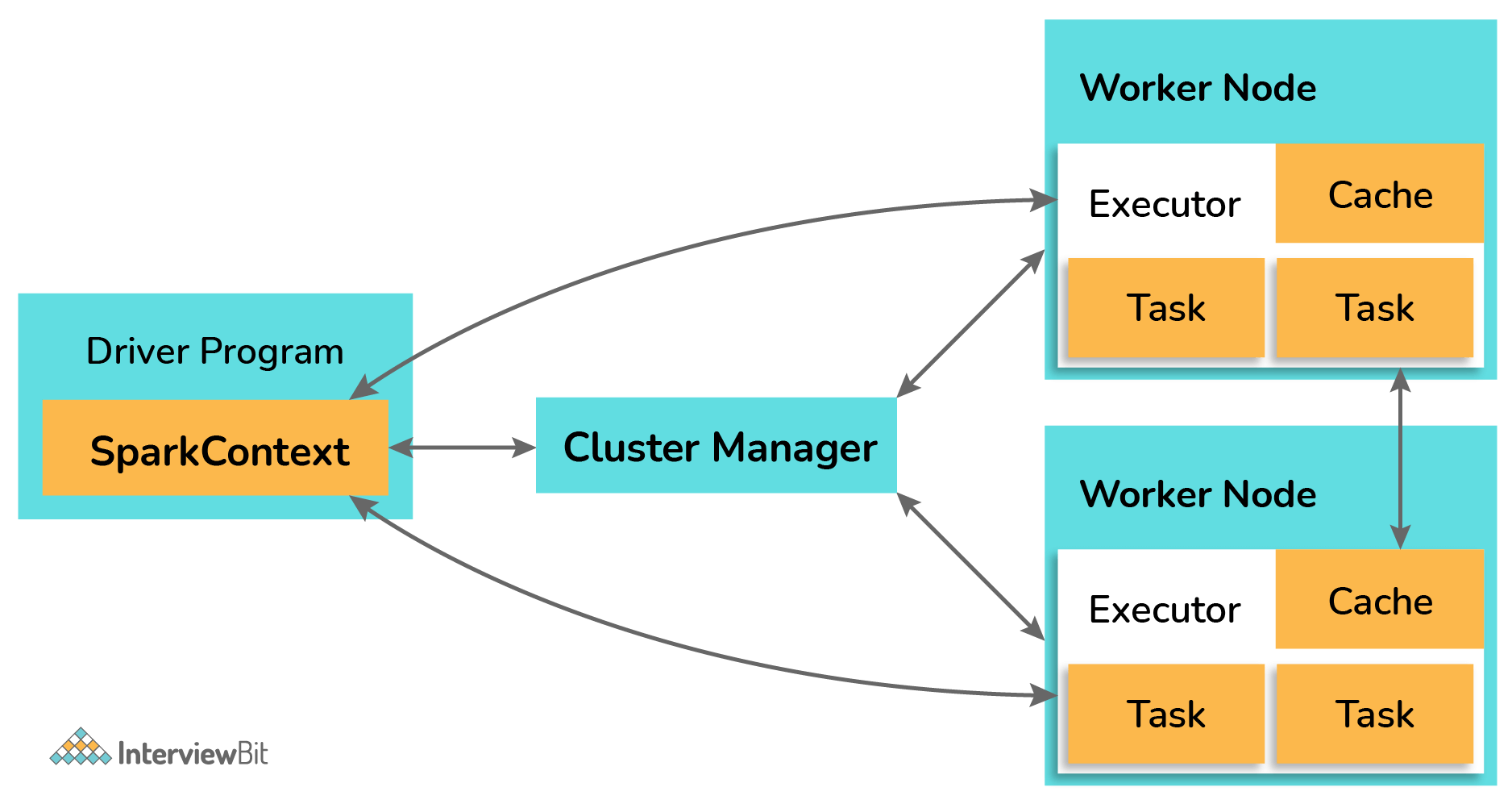
### 4. What does DAG refer to in Apache Spark?

DAG stands for Directed Acyclic Graph with no directed cycles. There would be finite vertices and edges. Each edge from one vertex is directed to another vertex in a *sequential manner*. The vertices refer to the RDDs of Spark and the edges represent the operations to be performed on those RDDs.

### 5. List the types of Deploy Modes in Spark.

There are 2 deploy modes in Spark. They are:

* **Client Mode**: The deploy mode is said to be in client mode when the spark driver component runs on the machine node from where the spark job is submitted.
  + The main disadvantage of this mode is if the machine node fails, then the entire job fails.
  + This mode supports both interactive shells or the job submission commands.
  + The performance of this mode is worst and is not preferred in production environments.
* **Cluster Mode**: If the spark job driver component does not run on the machine from which the spark job has been submitted, then the deploy mode is said to be in cluster mode.
  + The spark job launches the driver component within the cluster as a part of the sub-process of ApplicationMaster.
  + This mode supports deployment only using the spark-submit command (interactive shell mode is not supported).
  + Here, since the driver programs are run in ApplicationMaster, in case the program fails, the driver program is re-instantiated.
  + In this mode, there is a dedicated cluster manager (such as stand-alone, YARN, Apache Mesos, Kubernetes, etc) for allocating the resources required for the job to run as shown in the below architecture.



Apart from the above two modes, if we have to run the application on our local machines for unit testing and development, the deployment mode is called “**Local Mode**”. Here, the jobs run on a single JVM in a single machine which makes it highly inefficient as at some point or the other there would be a shortage of resources which results in the failure of jobs. It is also not possible to scale up resources in this mode due to the restricted memory and space.

### 6. What are receivers in Apache Spark Streaming?

Receivers are those entities that consume data from different data sources and then move them to Spark for processing. They are created by using streaming contexts in the form of long-running tasks that are scheduled for operating in a round-robin fashion. Each receiver is configured to use up only a single core. The receivers are made to run on various executors to accomplish the task of data streaming. There are two types of receivers depending on how the data is sent to Spark:

* **Reliable receivers**: Here, the receiver sends an acknowledegment to the data sources post successful reception of data and its replication on the Spark storage space.
* **Unreliable receiver**: Here, there is no acknowledgement sent to the data sources.

### 7. What is the difference between repartition and coalesce?

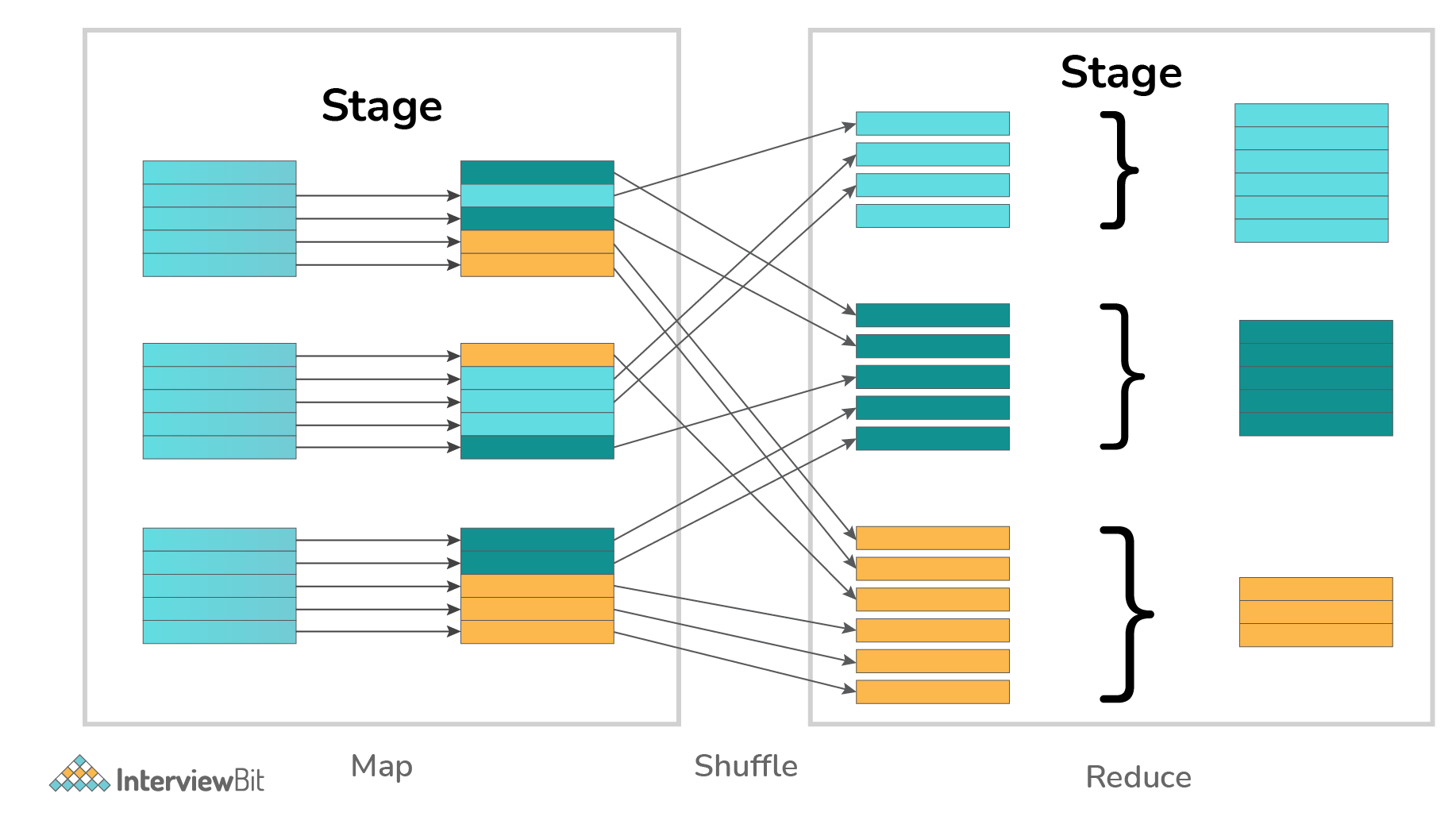
| **Repartition** | **Coalesce** |
| --- | --- |
| Usage repartition can increase/decrease the number of data partitions. | Spark coalesce can only reduce the number of data partitions. |
| Repartition creates new data partitions and performs a full shuffle of evenly distributed data. | Coalesce makes use of already existing partitions to reduce the amount of shuffled data unevenly. |
| Repartition internally calls coalesce with shuffle parameter thereby making it slower than coalesce. | Coalesce is faster than repartition. However, if there are unequal-sized data partitions, the speed might be slightly slower. |

### 8. What are the data formats supported by Spark?

Spark supports both the raw files and the structured file formats for efficient reading and processing. File formats like paraquet, JSON, XML, CSV, RC, Avro, TSV, etc are supported by Spark.

### 9. What do you understand by Shuffling in Spark?

The process of redistribution of data across different partitions which might or might not cause data movement across the JVM processes or the executors on the separate machines is known as shuffling/repartitioning. Partition is nothing but a smaller logical division of data.



It is to be noted that Spark has no control over what partition the data gets distributed across.

### 10. What is YARN in Spark?

* YARN is one of the key features provided by Spark that provides a central resource management platform for delivering scalable operations throughout the cluster.
* YARN is a cluster management technology and a Spark is a tool for data processing.

**Scaler Courses**

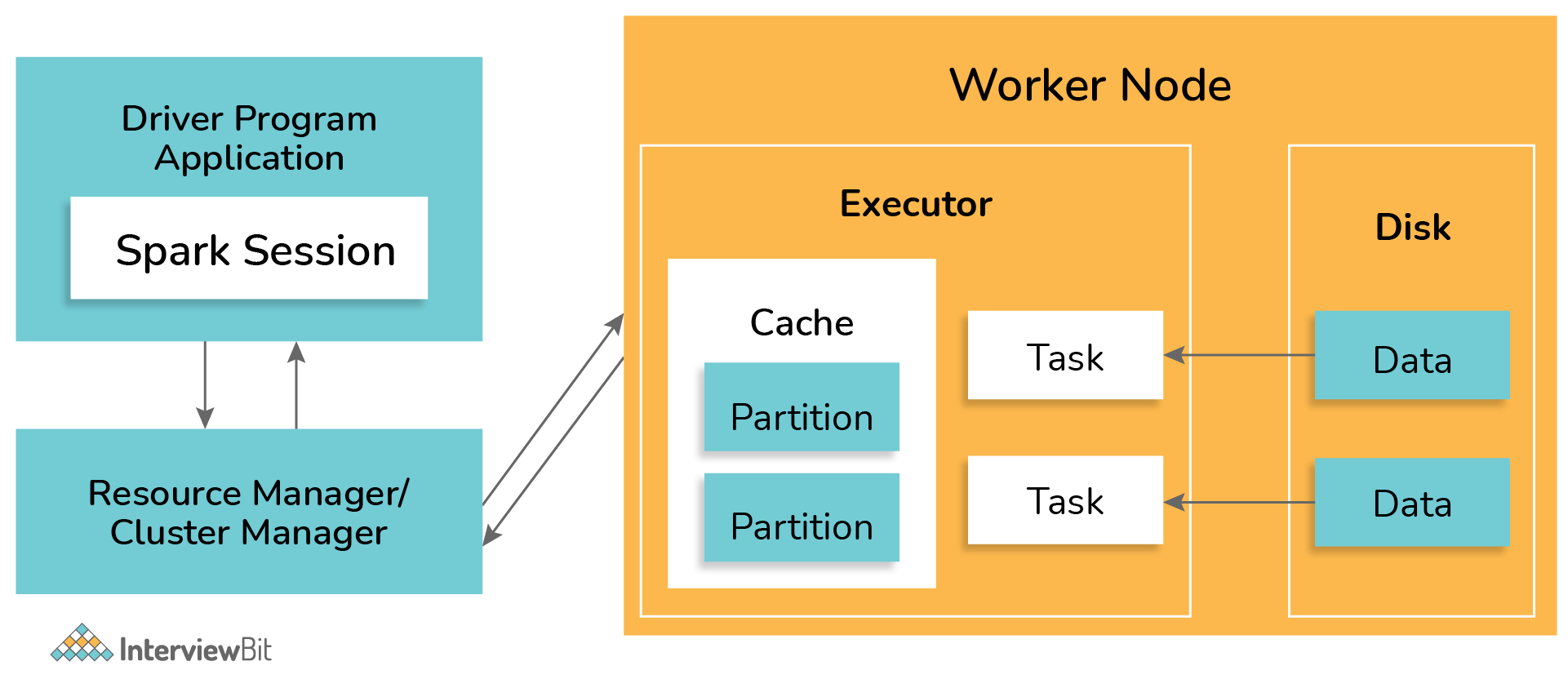
## Spark Interview Questions for Experienced

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

| **MapReduce** | **Apache Spark** |
| --- | --- |
| MapReduce does only batch-wise processing of data. | Apache Spark can process the data both in real-time and in batches. |
| MapReduce does slow processing of large data. | Apache Spark runs approximately 100 times faster than MapReduce for big data processing. |
| MapReduce stores data in HDFS (Hadoop Distributed File System) which makes it take a long time to get the data. | Spark stores data in memory (RAM) which makes it easier and faster to retrieve data when needed. |
| MapReduce highly depends on disk which makes it to be a high latency framework. | Spark supports in-memory data storage and caching and makes it a low latency computation framework. |
| MapReduce requires an external scheduler for jobs. | Spark has its own job scheduler due to the in-memory data computation. |

### 2. Explain the working of Spark with the help of its architecture.

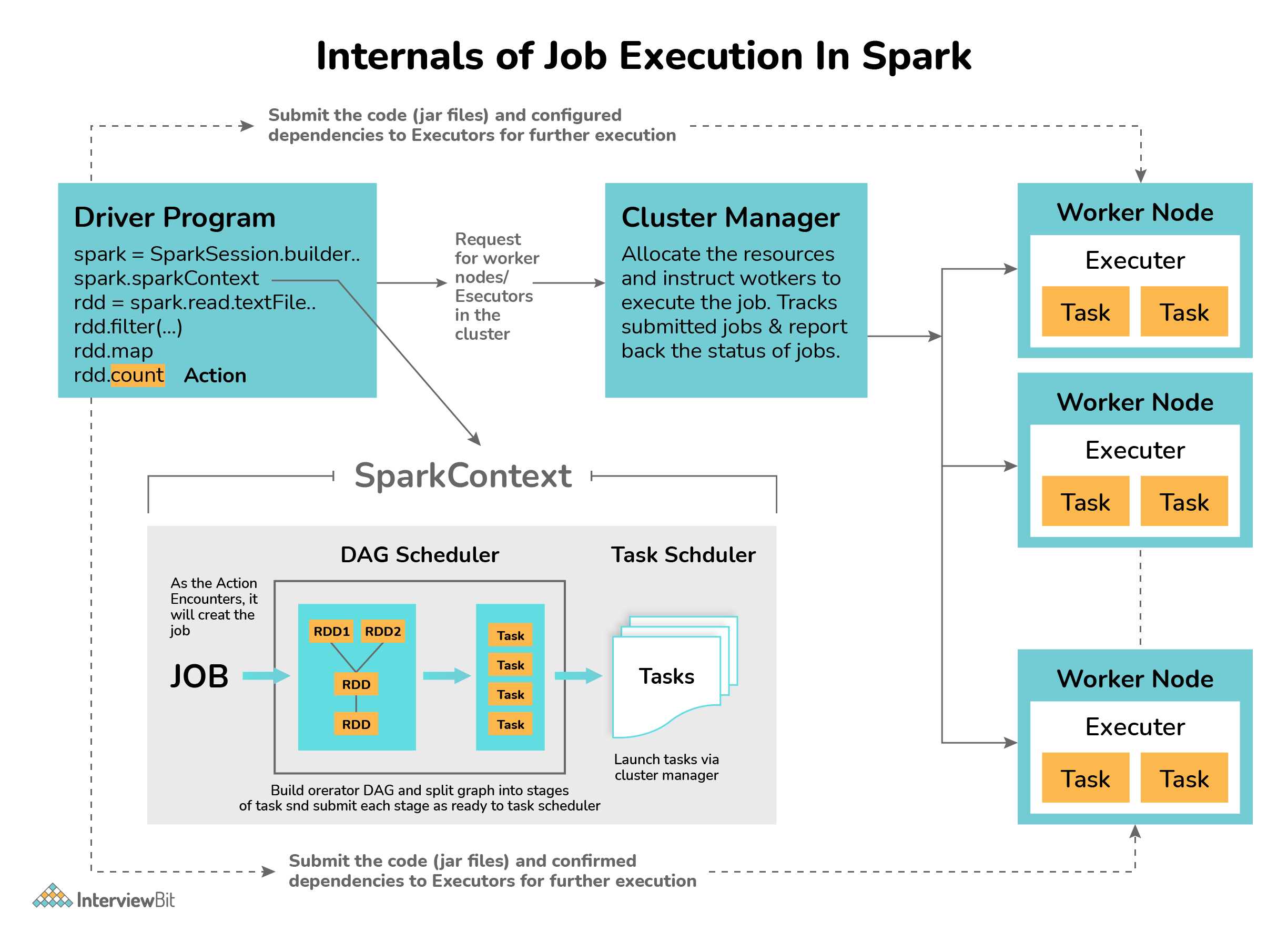
Spark applications are run in the form of independent processes that are well coordinated by the Driver program by means of a SparkSession object. The cluster manager or the resource manager entity of Spark assigns the tasks of running the Spark jobs to the worker nodes as per one task per partition principle. There are various iterations algorithms that are repeatedly applied to the data to cache the datasets across various iterations. Every task applies its unit of operations to the dataset within its partition and results in the new partitioned dataset. These results are sent back to the main driver application for further processing or to store the data on the disk. The following diagram illustrates this working as described above:



### 3. What is the working of DAG in Spark?

DAG stands for Direct Acyclic Graph which has a set of finite vertices and edges. The vertices represent RDDs and the edges represent the operations to be performed on RDDs sequentially. The DAG created is submitted to the DAG Scheduler which splits the graphs into stages of tasks based on the transformations applied to the data. The stage view has the details of the RDDs of that stage.

The working of DAG in spark is defined as per the workflow diagram below:



* The first task is to interpret the code with the help of an interpreter. If you use the Scala code, then the Scala interpreter interprets the code.
* Spark then creates an operator graph when the code is entered in the Spark console.
* When the action is called on Spark RDD, the operator graph is submitted to the DAG Scheduler.
* The operators are divided into stages of task by the DAG Scheduler. The stage consists of detailed step-by-step operation on the input data. The operators are then pipelined together.
* The stages are then passed to the Task Scheduler which launches the task via the cluster manager to work on independently without the dependencies between the stages.
* The worker nodes then execute the task.

Each RDD keeps track of the pointer to one/more parent RDD along with its relationship with the parent. For example, consider the operation val childB=parentA.map() on RDD, then we have the RDD childB that keeps track of its parentA which is called **RDD lineage**.

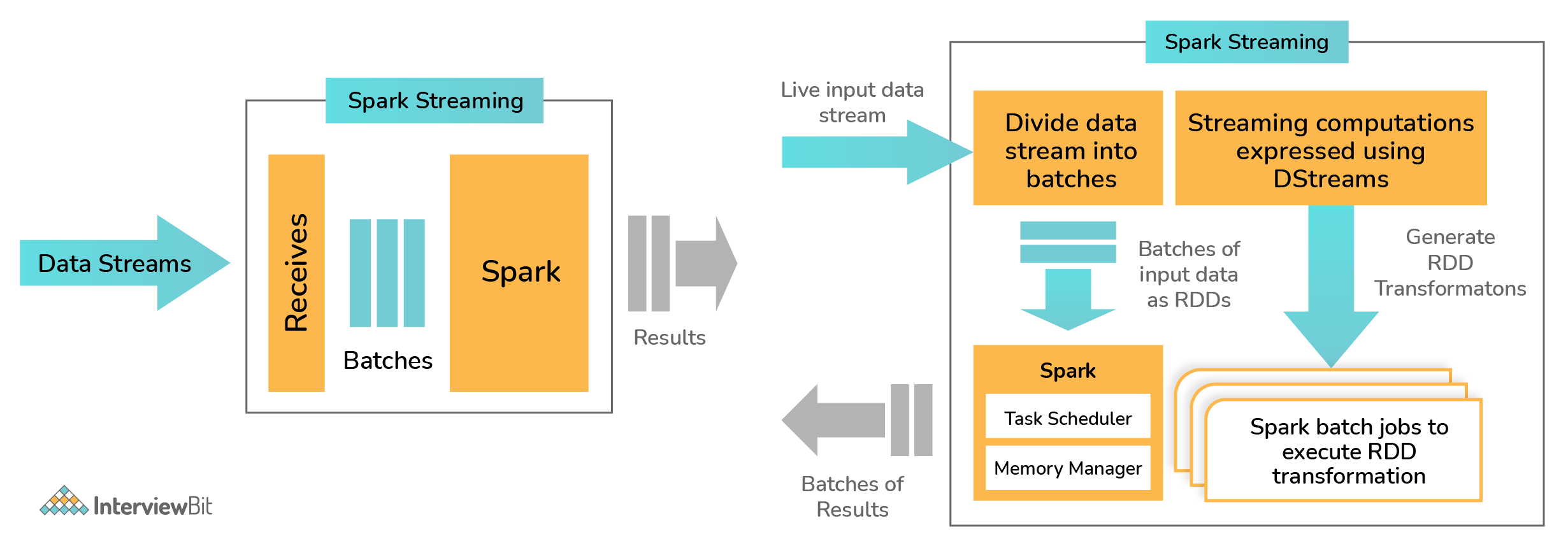
### 4. Under what scenarios do you use Client and Cluster modes for deployment?

* In case the client machines are not close to the cluster, then the Cluster mode should be used for deployment. This is done to avoid the network latency caused while communication between the executors which would occur in the Client mode. Also, in Client mode, the entire process is lost if the machine goes offline.
* If we have the client machine inside the cluster, then the Client mode can be used for deployment. Since the machine is inside the cluster, there won’t be issues of network latency and since the maintenance of the cluster is already handled, there is no cause of worry in cases of failure.

### 5. What is Spark Streaming and how is it implemented in Spark?

Spark Streaming is one of the most important features provided by Spark. It is nothing but a Spark API extension for supporting stream processing of data from different sources.

* Data from sources like Kafka, Kinesis, Flume, etc are processed and pushed to various destinations like databases, dashboards, machine learning APIs, or as simple as file systems. The data is divided into various streams (similar to batches) and is processed accordingly.
* Spark streaming supports highly scalable, fault-tolerant continuous stream processing which is mostly used in cases like fraud detection, website monitoring, website click baits, IoT (Internet of Things) sensors, etc.
* Spark Streaming first divides the data from the data stream into batches of X seconds which are called Dstreams or Discretized Streams. They are internally nothing but a sequence of multiple RDDs. The Spark application does the task of processing these RDDs using various Spark APIs and the results of this processing are again returned as batches. The following diagram explains the workflow of the spark streaming process.



### 6. Write a spark program to check if a given keyword exists in a huge text file or not?

def keywordExists(line):

if (line.find(“my\_keyword”) > -1):

return 1

return 0

lines = sparkContext.textFile(“test\_file.txt”);

isExist = lines.map(keywordExists);

sum = isExist.reduce(sum);

print(“Found” if sum>0 else “Not Found”)

### 7. What can you say about Spark Datasets?

Spark Datasets are those data structures of SparkSQL that provide JVM objects with all the benefits (such as data manipulation using lambda functions) of RDDs alongside Spark SQL-optimised execution engine. This was introduced as part of Spark since version 1.6.

* Spark datasets are strongly typed structures that represent the structured queries along with their encoders.
* They provide type safety to the data and also give an object-oriented programming interface.
* The datasets are more structured and have the lazy query expression which helps in triggering the action. Datasets have the combined powers of both RDD and Dataframes. Internally, each dataset symbolizes a logical plan which informs the computational query about the need for data production. Once the logical plan is analyzed and resolved, then the physical query plan is formed that does the actual query execution.

Datasets have the following features:

* **Optimized Query feature**: Spark datasets provide optimized queries using Tungsten and Catalyst Query Optimizer frameworks. The Catalyst Query Optimizer represents and manipulates a data flow graph (graph of expressions and relational operators). The Tungsten improves and optimizes the speed of execution of Spark job by emphasizing the hardware architecture of the Spark execution platform.
* **Compile-Time Analysis**: Datasets have the flexibility of analyzing and checking the syntaxes at the compile-time which is not technically possible in RDDs or Dataframes or the regular SQL queries.
* **Interconvertible**: The type-safe feature of datasets can be converted to “untyped” Dataframes by making use of the following methods provided by the Datasetholder:
  + toDS():Dataset[T]
  + toDF():DataFrame
  + toDF(columName:String\*):DataFrame
* **Faster Computation:** Datasets implementation are much faster than those of the RDDs which helps in increasing the system performance.
* **Persistent storage qualified**: Since the datasets are both queryable and serializable, they can be easily stored in any persistent storages.
* **Less Memory Consumed**: Spark uses the feature of caching to create a more optimal data layout. Hence, less memory is consumed.
* **Single Interface Multiple Languages**: Single API is provided for both Java and Scala languages. These are widely used languages for using Apache Spark. This results in a lesser burden of using libraries for different types of inputs.

### 8. Define Spark DataFrames.

Spark Dataframes are the distributed collection of datasets organized into columns similar to SQL. It is equivalent to a table in the relational database and is mainly optimized for big data operations.  
Dataframes can be created from an array of data from different data sources such as external databases, existing RDDs, Hive Tables, etc. Following are the features of Spark Dataframes:

* Spark Dataframes have the ability of processing data in sizes ranging from Kilobytes to Petabytes on a single node to large clusters.
* They support different data formats like CSV, Avro, elastic search, etc, and various storage systems like HDFS, Cassandra, MySQL, etc.
* By making use of SparkSQL catalyst optimizer, state of art optimization is achieved.
* It is possible to easily integrate Spark Dataframes with major Big Data tools using SparkCore.

### 9. Define Executor Memory in Spark

The applications developed in Spark have the same fixed cores count and fixed heap size defined for spark executors. The heap size refers to the memory of the Spark executor that is controlled by making use of the property spark.executor.memory that belongs to the -executor-memory flag. Every Spark applications have one allocated executor on each worker node it runs. The executor memory is a measure of the memory consumed by the worker node that the application utilizes.

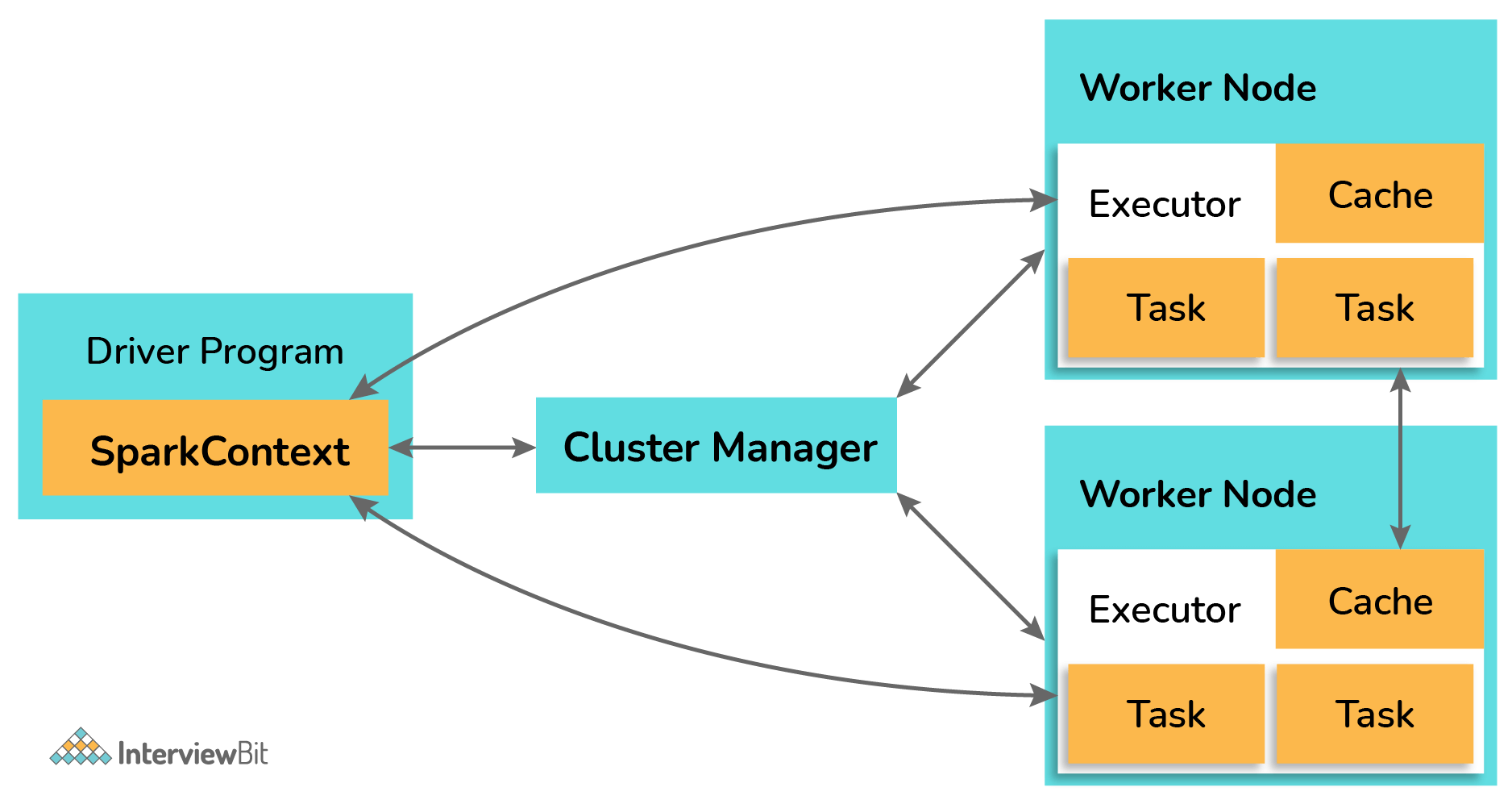
### 10. What are the functions of SparkCore?

SparkCore is the main engine that is meant for large-scale distributed and parallel data processing. The Spark core consists of the distributed execution engine that offers various APIs in Java, Python, and Scala for developing distributed ETL applications.  
Spark Core does important functions such as memory management, job monitoring, fault-tolerance, storage system interactions, job scheduling, and providing support for all the basic I/O functionalities. There are various additional libraries built on top of Spark Core which allows diverse workloads for SQL, streaming, and machine learning. They are responsible for:

* Fault recovery
* Memory management and Storage system interactions
* Job monitoring, scheduling, and distribution
* Basic I/O functions

### 11. What do you understand by worker node?

Worker nodes are those nodes that run the Spark application in a cluster. The Spark driver program listens for the incoming connections and accepts them from the executors addresses them to the worker nodes for execution. A worker node is like a slave node where it gets the work from its master node and actually executes them. The worker nodes do data processing and report the resources used to the master. The master decides what amount of resources needs to be allocated and then based on their availability, the tasks are scheduled for the worker nodes by the master.



### 12. What are some of the demerits of using Spark in applications?

Despite Spark being the powerful data processing engine, there are certain demerits to using Apache Spark in applications. Some of them are:

* Spark makes use of more storage space when compared to MapReduce or Hadoop which may lead to certain memory-based problems.
* Care must be taken by the developers while running the applications. The work should be distributed across multiple clusters instead of running everything on a single node.
* Since Spark makes use of “in-memory” computations, they can be a bottleneck to cost-efficient big data processing.
* While using files present on the path of the local filesystem, the files must be accessible at the same location on all the worker nodes when working on cluster mode as the task execution shuffles between various worker nodes based on the resource availabilities. The files need to be copied on all worker nodes or a separate network-mounted file-sharing system needs to be in place.
* One of the biggest problems while using Spark is when using a large number of small files. When Spark is used with Hadoop, we know that HDFS gives a limited number of large files instead of a large number of small files. When there is a large number of small gzipped files, Spark needs to uncompress these files by keeping them on its memory and network. So large amount of time is spent in burning core capacities for unzipping the files in sequence and performing partitions of the resulting RDDs to get data in a manageable format which would require extensive shuffling overall. This impacts the performance of Spark as much time is spent preparing the data instead of processing them.
* Spark doesn’t work well in multi-user environments as it is not capable of handling many users concurrently.

### 13. How can the data transfers be minimized while working with Spark?

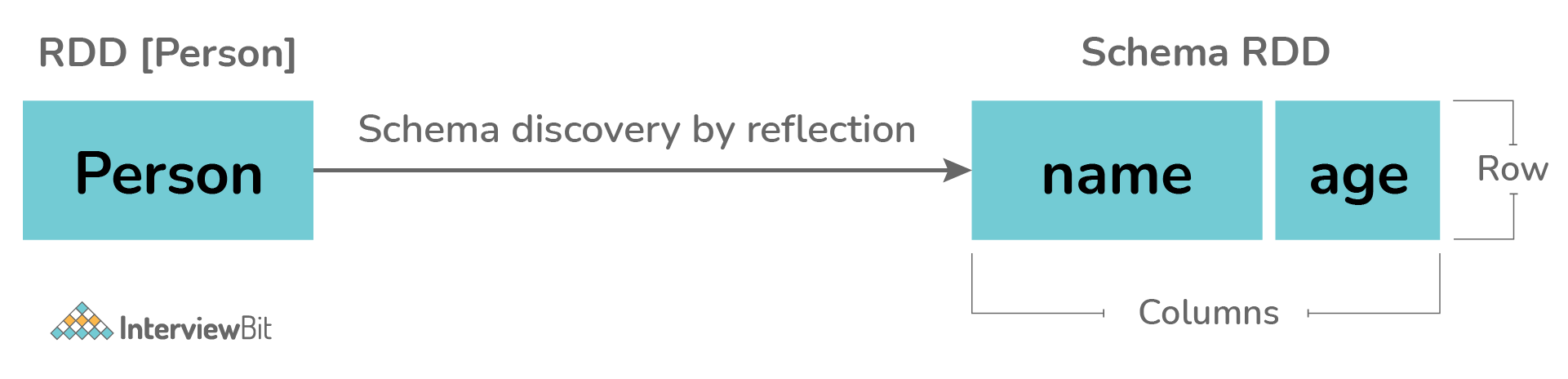
Data transfers correspond to the process of shuffling. Minimizing these transfers results in faster and reliable running Spark applications. There are various ways in which these can be minimized. They are:

* Usage of Broadcast Variables: Broadcast variables increases the efficiency of the join between large and small RDDs.
* Usage of Accumulators: These help to update the variable values parallelly during execution.
* Another common way is to avoid the operations which trigger these reshuffles.

### 14. What is SchemaRDD in Spark RDD?

SchemaRDD is an RDD consisting of row objects that are wrappers around integer arrays or strings that has schema information regarding the data type of each column. They were designed to ease the lives of developers while debugging the code and while running unit test cases on the SparkSQL modules. They represent the description of the RDD which is similar to the schema of relational databases. SchemaRDD also provides the basic functionalities of the common RDDs along with some relational query interfaces of SparkSQL.

Consider an example. If you have an RDD named Person that represents a person’s data. Then SchemaRDD represents what data each row of Person RDD represents. If the Person has attributes like name and age, then they are represented in SchemaRDD.



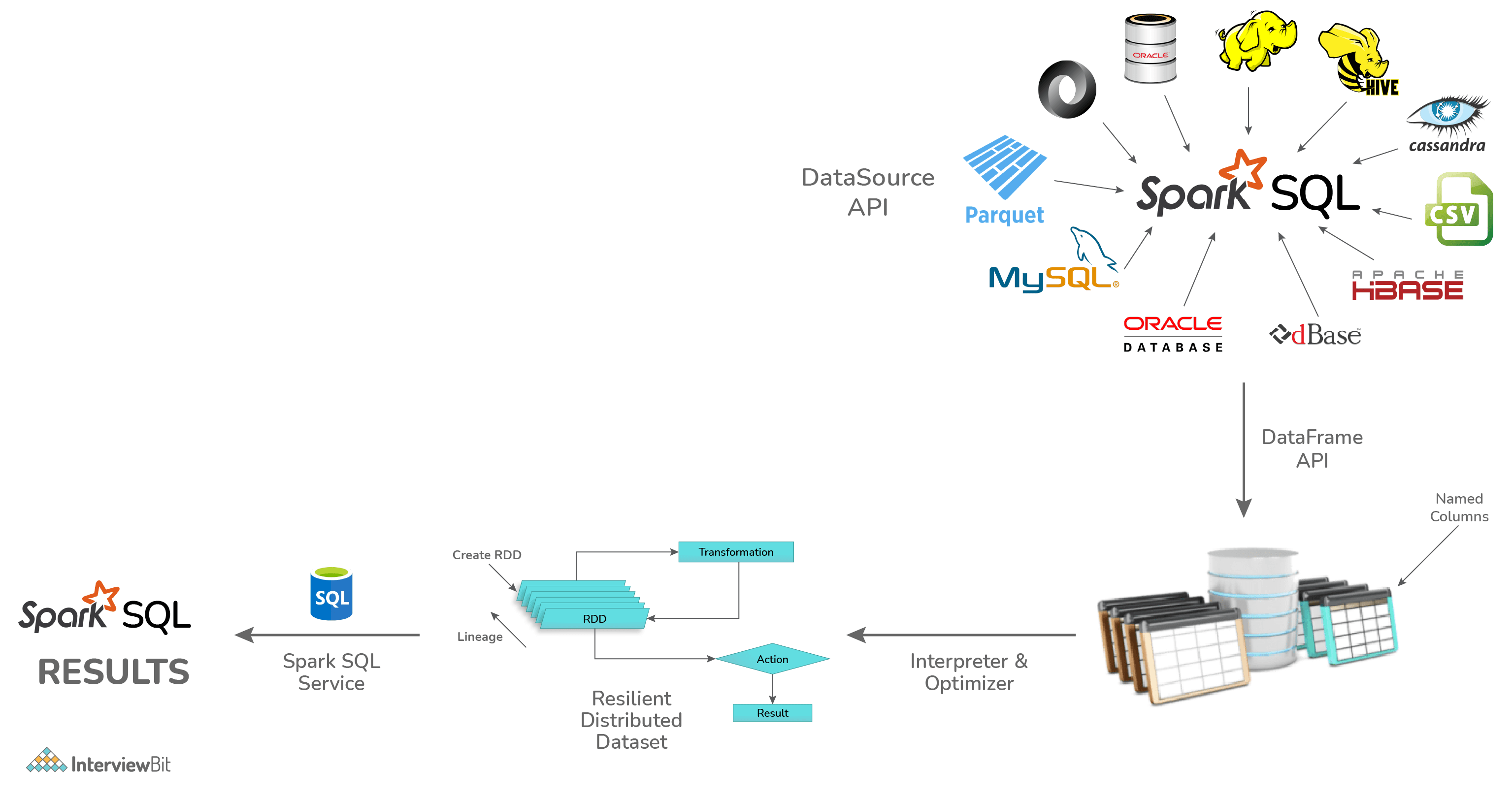
### 15. What module is used for implementing SQL in Apache Spark?

Spark provides a powerful module called SparkSQL which performs relational data processing combined with the power of the functional programming feature of Spark. This module also supports either by means of SQL or Hive Query Language. It also provides support for different data sources and helps developers write powerful SQL queries using code transformations.  
The four major libraries of SparkSQL are:

* Data Source API
* DataFrame API
* Interpreter & Catalyst Optimizer
* SQL Services

Spark SQL supports the usage of structured and semi-structured data in the following ways:

* Spark supports DataFrame abstraction in various languages like Python, Scala, and Java along with providing good optimization techniques.
* SparkSQL supports data read and writes operations in various structured formats like JSON, Hive, Parquet, etc.
* SparkSQL allows data querying inside the Spark program and via external tools that do the JDBC/ODBC connections.
* It is recommended to use SparkSQL inside the Spark applications as it empowers the developers to load the data, query the data from databases and write the results to the destination.



### 16. What are the different persistence levels in Apache Spark?

Spark persists intermediary data from different shuffle operations automatically. But it is recommended to call the persist() method on the RDD. There are different persistence levels for storing the RDDs on memory or disk or both with different levels of replication. The persistence levels available in Spark are:

* **MEMORY\_ONLY**: This is the default persistence level and is used for storing the RDDs as the deserialized version of Java objects on the JVM. In case the RDDs are huge and do not fit in the memory, then the partitions are not cached and they will be recomputed as and when needed.
* **MEMORY\_AND\_DISK**: The RDDs are stored again as deserialized Java objects on JVM. In case the memory is insufficient, then partitions not fitting on the memory will be stored on disk and the data will be read from the disk as and when needed.
* **MEMORY\_ONLY\_SER**: The RDD is stored as serialized Java Objects as One Byte per partition.
* **MEMORY\_AND\_DISK\_SER**: This level is similar to MEMORY\_ONLY\_SER but the difference is that the partitions not fitting in the memory are saved on the disk to avoid recomputations on the fly.
* **DISK\_ONLY**: The RDD partitions are stored only on the disk.
* **OFF\_HEAP**: This level is the same as the MEMORY\_ONLY\_SER but here the data is stored in the off-heap memory.

The syntax for using persistence levels in the persist() method is:

df.persist(StorageLevel.<level\_value>)

The following table summarizes the details of persistence levels:

| **Persistence Level** | **Space Consumed** | **CPU time** | **In-memory?** | **On-disk?** |
| --- | --- | --- | --- | --- |
| MEMORY\_ONLY | High | Low | Yes | No |
| MEMORY\_ONLY\_SER | Low | High | Yes | No |
| MEMORY\_AND\_DISK | High | Medium | Some | Some |
| MEMORY\_AND\_DISK\_SER | Low | High | Some | Some |
| DISK\_ONLY | Low | High | No | Yes |
| OFF\_HEAP | Low | High | Yes (but off-heap) | No |

### 17. What are the steps to calculate the executor memory?

Consider you have the below details regarding the cluster:

Number of nodes = 10

Number of cores in each node = 15 cores

RAM of each node = 61GB

To identify the number of cores, we follow the approach:

Number of Cores = number of concurrent tasks that can be run parallelly by the executor. The optimal value as part of a general rule of thumb is 5.

Hence to calculate the number of executors, we follow the below approach:

Number of executors = Number of cores/Concurrent Task

= 15/5

= 3

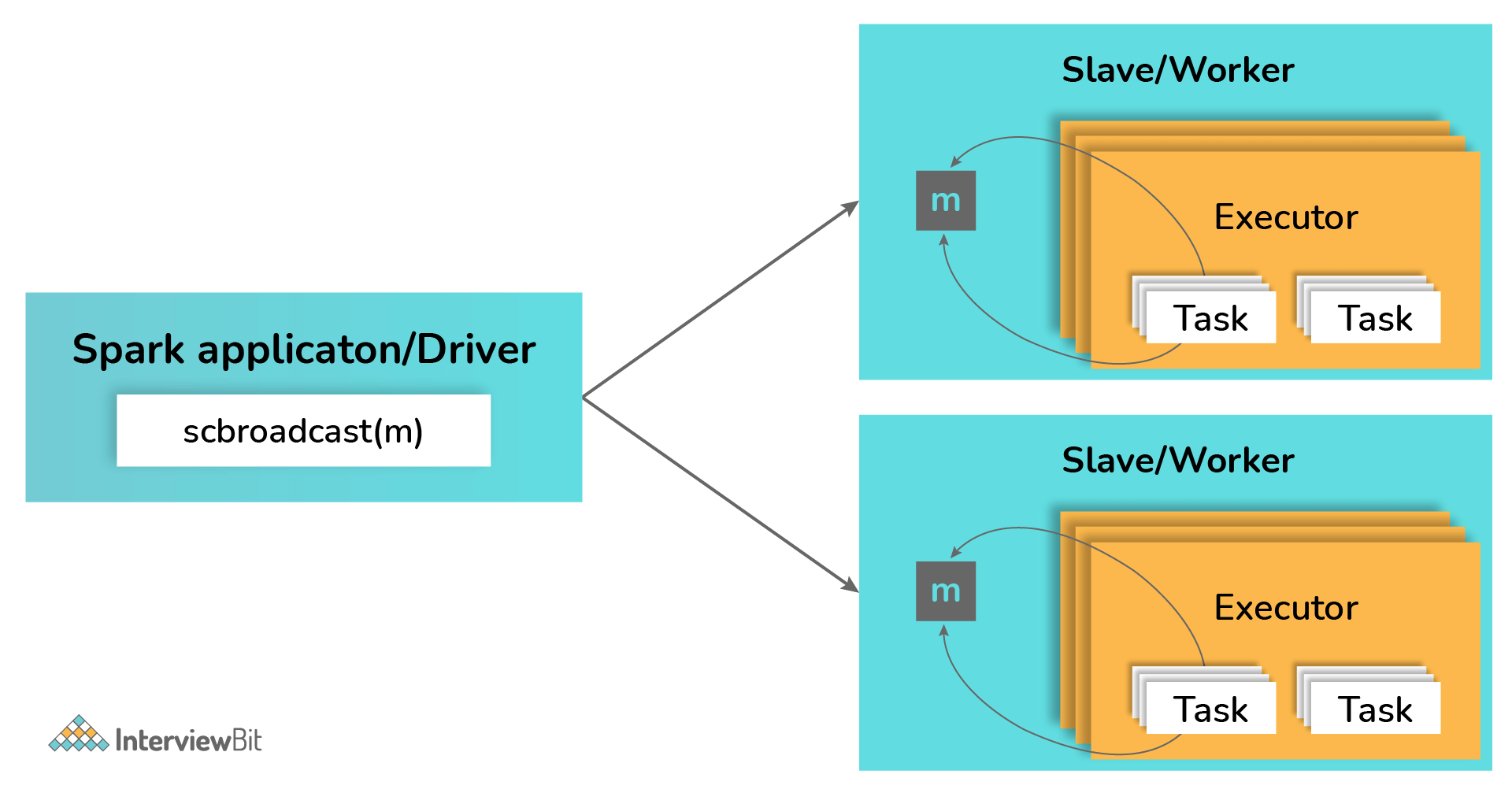
Number of executors = Number of nodes \* Number of executor in each node

= 10 \* 3

= 30 executors per Spark job

### 18. Why do we need broadcast variables in Spark?

Broadcast variables let the developers maintain read-only variables cached on each machine instead of shipping a copy of it with tasks. They are used to give every node copy of a large input dataset efficiently. These variables are broadcasted to the nodes using different algorithms to reduce the cost of communication.

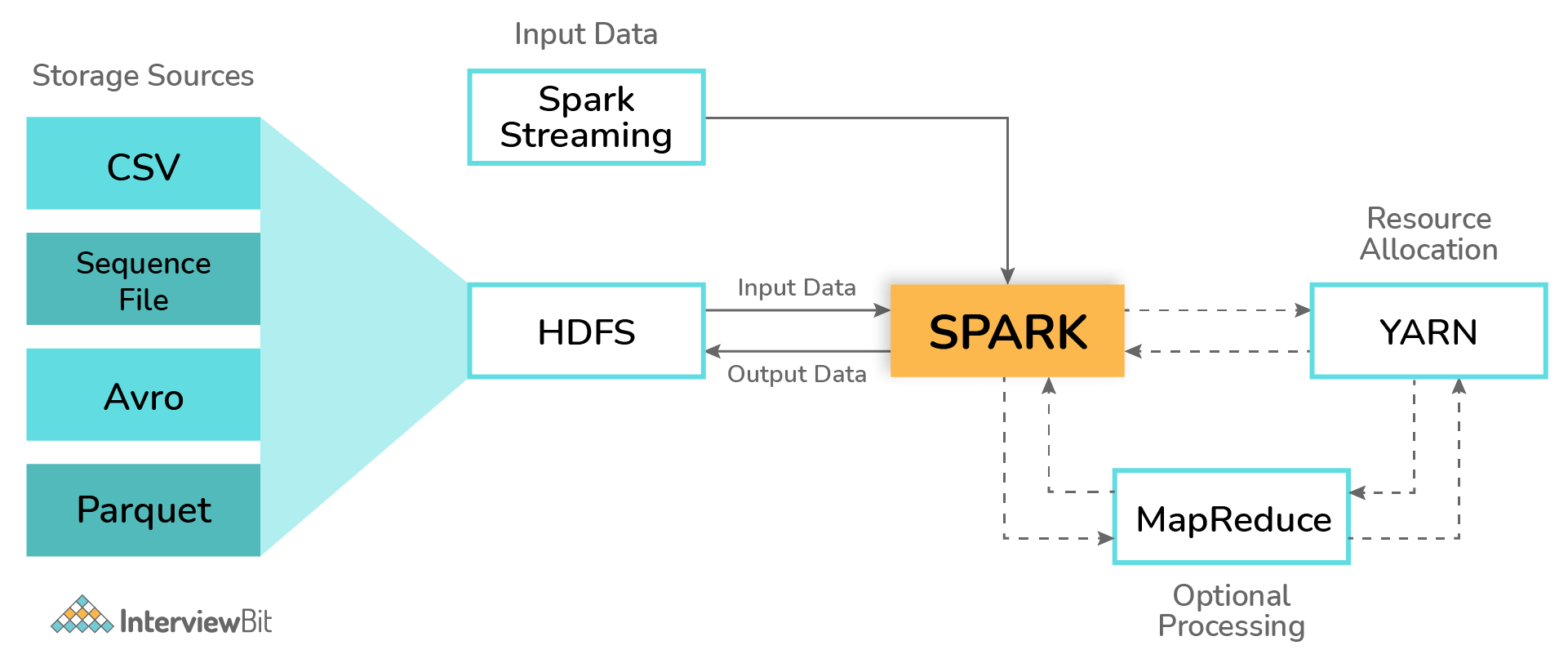


### 19. Differentiate between Spark Datasets, Dataframes and RDDs.

| **Criteria** | **Spark Datasets** | **Spark Dataframes** | **Spark RDDs** |
| --- | --- | --- | --- |
| **Representation of Data** | Spark Datasets is a combination of Dataframes and RDDs with features like static type safety and object-oriented interfaces. | Spark Dataframe is a distributed collection of data that is organized into named columns. | Spark RDDs are a distributed collection of data without schema. |
| **Optimization** | Datasets make use of catalyst optimizers for optimization. | Dataframes also makes use of catalyst optimizer for optimization. | There is no built-in optimization engine. |
| **Schema Projection** | Datasets find out schema automatically using SQL Engine. | Dataframes also find the schema automatically. | Schema needs to be defined manually in RDDs. |
| **Aggregation Speed** | Dataset aggregation is faster than RDD but slower than Dataframes. | Aggregations are faster in Dataframes due to the provision of easy and powerful APIs. | RDDs are slower than both the Dataframes and the Datasets while performing even simple operations like data grouping. |

### 20. Can Apache Spark be used along with Hadoop? If yes, then how?

Yes! The main feature of Spark is its compatibility with Hadoop. This makes it a powerful framework as using the combination of these two helps to leverage the processing capacity of Spark by making use of the best of Hadoop’s YARN and HDFS features.



Hadoop can be integrated with Spark in the following ways:

* **HDFS**: Spark can be configured to run atop HDFS to leverage the feature of distributed replicated storage.
* **MapReduce**: Spark can also be configured to run alongside the MapReduce in the same or different processing framework or Hadoop cluster. Spark and MapReduce can be used together to perform real-time and batch processing respectively.
* **YARN**: Spark applications can be configured to run on YARN which acts as the cluster management framework.

### 21. What are Sparse Vectors? How are they different from dense vectors?

Sparse vectors consist of two parallel arrays where one array is for storing indices and the other for storing values. These vectors are used to store non-zero values for saving space.

val sparseVec: Vector = Vectors.sparse(5, Array(0, 4), Array(1.0, 2.0))

* In the above example, we have the vector of size 5, but the non-zero values are there only at indices 0 and 4.
* Sparse vectors are particularly useful when there are very few non-zero values. If there are cases that have only a few zero values, then it is recommended to use dense vectors as usage of sparse vectors would introduce the overhead of indices which could impact the performance.
* Dense vectors can be defines as follows:

val denseVec = Vectors.dense(4405d,260100d,400d,5.0,4.0,198.0,9070d,1.0,1.0,2.0,0.0)

* Usage of sparse or dense vectors does not impact the results of calculations but when used inappropriately, they impact the memory consumed and the speed of calculation.

### 22. How are automatic clean-ups triggered in Spark for handling the accumulated metadata?

The clean-up tasks can be triggered automatically either by setting spark.cleaner.ttl parameter or by doing the batch-wise division of the long-running jobs and then writing the intermediary results on the disk.

### 23. How is Caching relevant in Spark Streaming?

Spark Streaming involves the division of data stream’s data into batches of X seconds called DStreams. These DStreams let the developers cache the data into the memory which can be very useful in case the data of DStream is used for multiple computations. The caching of data can be done using the cache() method or using persist() method by using appropriate persistence levels. The default persistence level value for input streams receiving data over the networks such as Kafka, Flume, etc is set to achieve data replication on 2 nodes to accomplish fault tolerance.

* Caching using cache method:

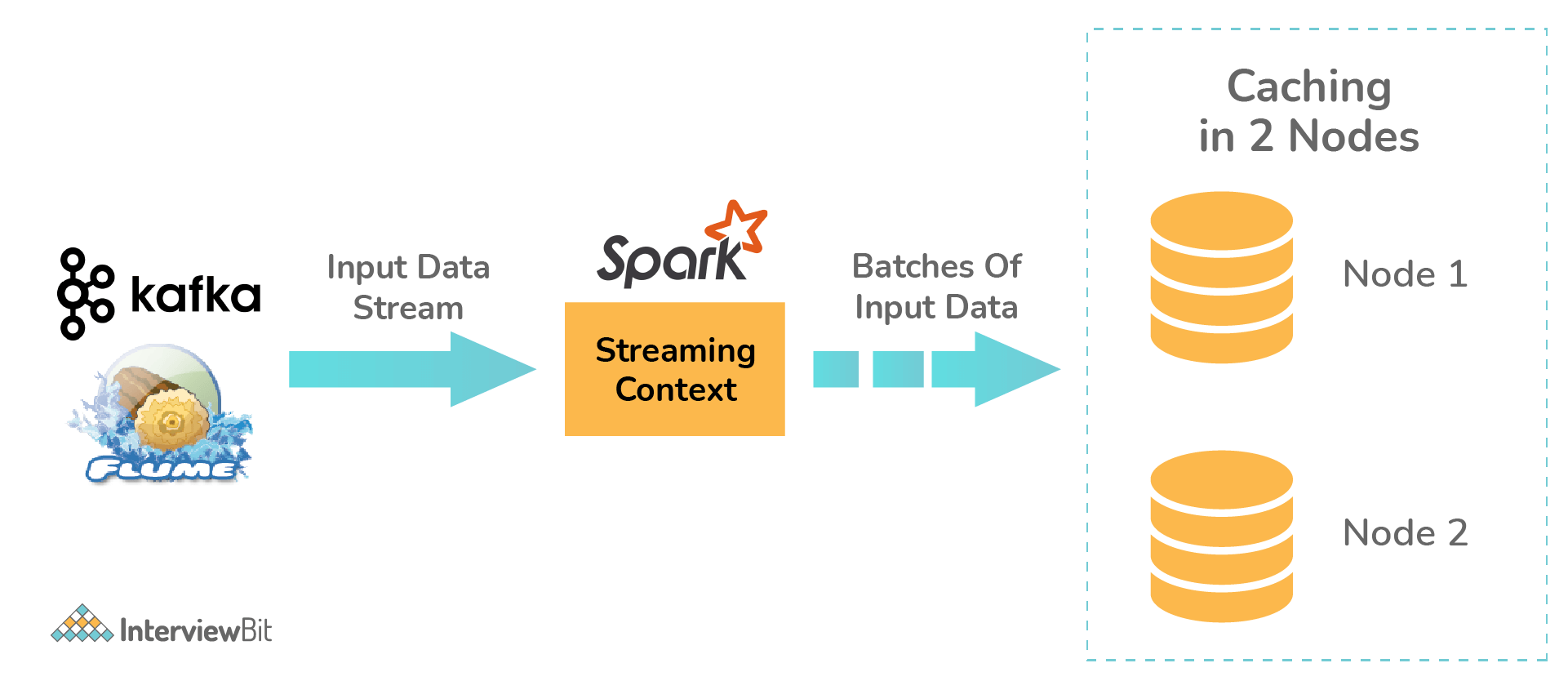
val cacheDf = dframe.cache()

* Caching using persist method:

val persistDf = dframe.persist(StorageLevel.MEMORY\_ONLY)

The main advantages of caching are:

* **Cost efficiency**: Since Spark computations are expensive, caching helps to achieve reusing of data and this leads to reuse computations which can save the cost of operations.
* **Time-efficient**: The computation reusage leads to saving a lot of time.
* **More Jobs Achieved**: By saving time of computation execution, the worker nodes can perform/execute more jobs.



### 24. Define Piping in Spark.

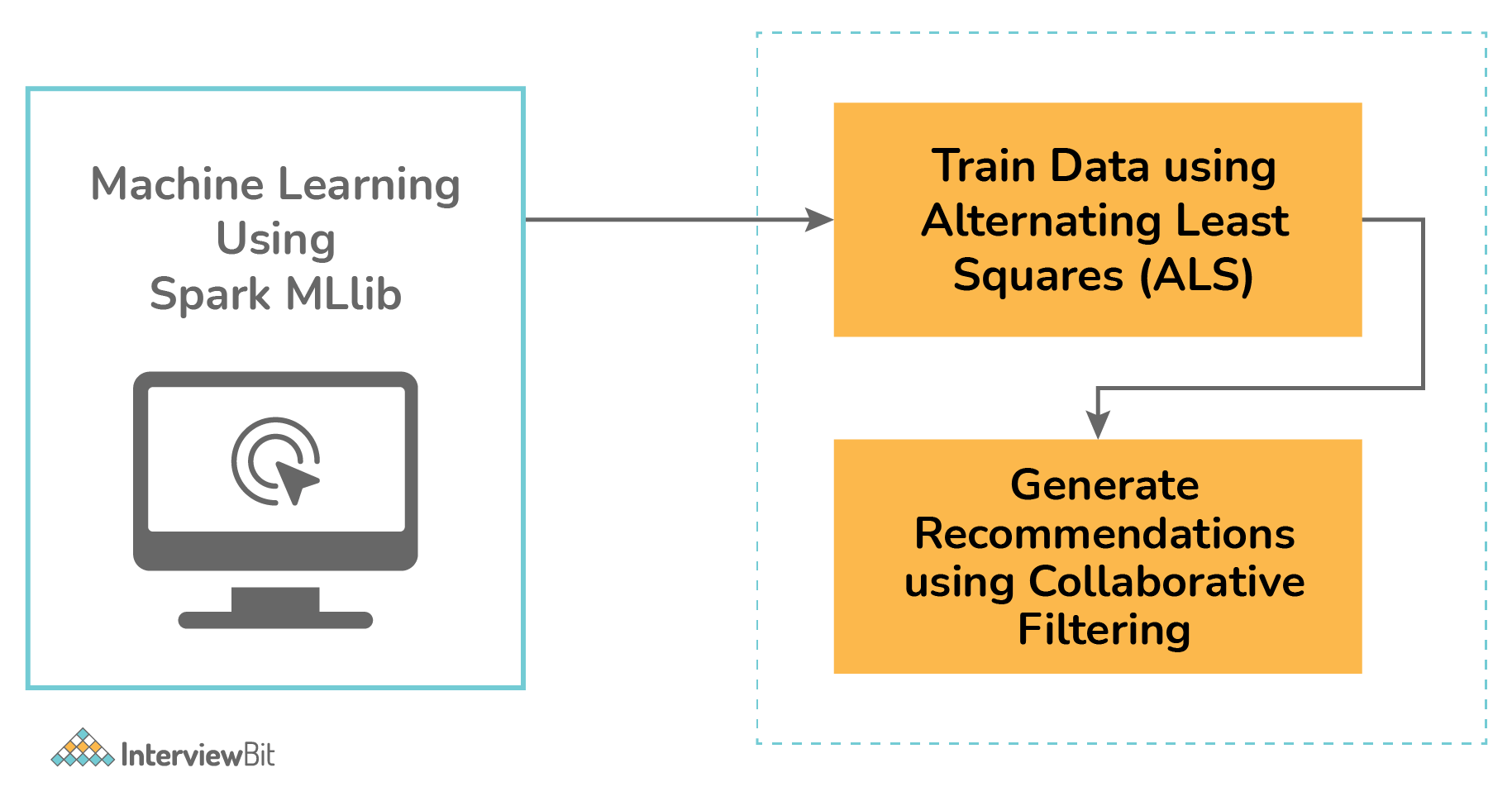
Apache Spark provides the pipe() method on RDDs which gives the opportunity to compose different parts of occupations that can utilize any language as needed as per the UNIX Standard Streams. Using the pipe() method, the RDD transformation can be written which can be used for reading each element of the RDD as String. These can be manipulated as required and the results can be displayed as String.

### 25. What API is used for Graph Implementation in Spark?

Spark provides a powerful API called GraphX that extends Spark RDD for supporting graphs and graph-based computations. The extended property of Spark RDD is called as Resilient Distributed Property Graph which is a directed multi-graph that has multiple parallel edges. Each edge and the vertex has associated user-defined properties. The presence of parallel edges indicates multiple relationships between the same set of vertices. GraphX has a set of operators such as subgraph, mapReduceTriplets, joinVertices, etc that can support graph computation. It also includes a large collection of graph builders and algorithms for simplifying tasks related to graph analytics.

### 26. How can you achieve machine learning in Spark?

Spark provides a very robust, scalable machine learning-based library called MLlib. This library aims at implementing easy and scalable common ML-based algorithms and has the features like classification, clustering, dimensional reduction, regression filtering, etc. More information about this library can be obtained in detail from Spark’s official documentation site here: <https://spark.apache.org/docs/latest/ml-guide.html>



### Conclusion

In this article, we have seen the most commonly asked Spark interview questions. Apache Spark is the fastest-growing cluster computational platform that was designed to process big data in a faster manner along with the compatibility to previously existing big data tools and support to various libraries. These integrations help to build seamlessly fast and powerful applications with the power of different computational models. Due to these reasons, Spark has become a hot and lucrative technology, and knowing Spark will open doors to new, better, and challenging career opportunities for Software Developers and Data Engineers.

## Top Answer to Apache Spark Interview Questions

As a professional in the field of [Big Data](https://intellipaat.com/blog/tutorial/big-data-and-hadoop-tutorial/introduction-to-big-data/), it is important for you to know all the terms and technologies related to this field, including Apache Spark, which is among the most popular and in-demand technologies in Big Data. Go through these Apache Spark interview questions to prepare for job interviews to get a head start in your[career in Big Data](https://intellipaat.com/blog/big-data-career-path/):

## Frequently asked Spark interview questions:

[Q1. What is Apache Spark?](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#1)  
[Q2. Explain the key features of Spark.](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#2)  
[Q3. What is MapReduce?](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#3)  
[Q4. Compare MapReduce with Spark.](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#4)  
[Q5. Define RDD.](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#5)  
[Q6. What does a Spark Engine do?](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#6)  
[Q7. Define Partitions.](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#7)  
[Q8. What operations does an RDD support?](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#8)  
[Q9. What do you understand about Transformations in Spark?](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#9)  
[Q10. Define Actions in Spark.](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#10)

These Apache Spark interview questions and answers are majorly classified into the following categories:  
[1. Basic interview questions](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#11)

[2. Intermediate interview questions](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#12)

[3. Advanced interview questions](https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#13)

## Basic Spark Interview Questions

### **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/), [Python](https://intellipaat.com/blog/tutorial/python-tutorial/what-is-python/), [Java](https://intellipaat.com/blog/tutorial/java-tutorial/), and [R](https://intellipaat.com/blog/tutorial/r-programming/introduction/). 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/).
* 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?**

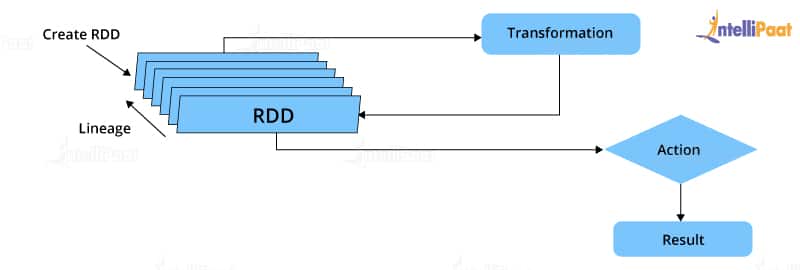
It is a software framework and programming model which is used for processing huge datasets. [MapReduce](https://intellipaat.com/blog/tutorial/mapreduce-tutorial/) is basically split into two parts, Map and Reduce. Map handles data splitting and data mapping, meanwhile, Reduce handles shuffle and reduction 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/)—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)***!***

**Get 100% Hike!**

Master Most in Demand Skills Now !

Top of Form



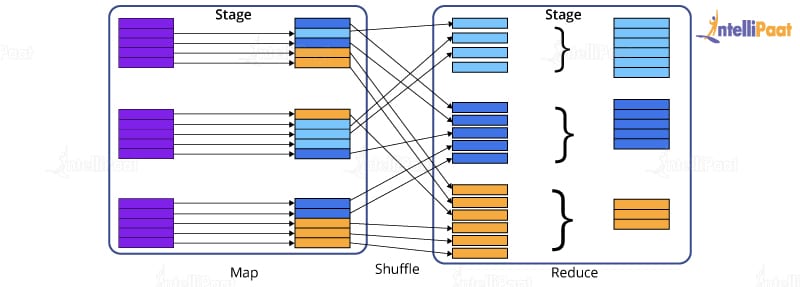


 By providing your contact details, you agree to our [Terms of Use](https://intellipaat.com/terms-of-use/) & [Privacy Policy](https://intellipaat.com/terms-of-use-and-privacy-policy/)

Bottom of Form

### **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.**

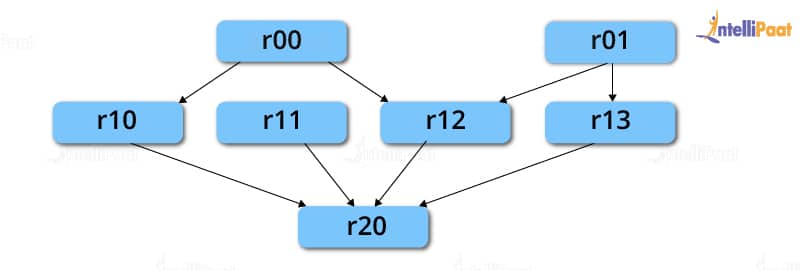
Serving as the base engine, Spark Core performs various important functions like memory management, basic I/O functionalities, monitoring jobs, providing fault-tolerance, job scheduling, interaction with storage systems, distributed task dispatching, and many more. Spark Core is the base of all projects. 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 Spark Interview Questions for experienced

### **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**

[[](javascript:void(0);)](javascript:void(0);)

[[](javascript:void(0);)](javascript:void(0);)

[[](javascript:void(0);)](javascript:void(0);)

### **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/) 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;

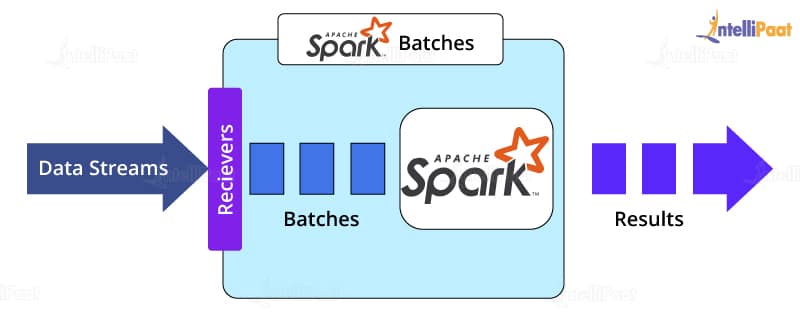
Hive supports Spark on YARN mode by default.

### **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/))
* 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/)***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/) 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/) formats so far.

**Courses you may like**

[](https://intellipaat.com/advanced-certification-data-science-artificial-intelligence-iit-madras/)

[](https://intellipaat.com/pg-certification-data-science-machine-learning-mnit/)

[](https://intellipaat.com/data-science-architect-masters-program-training/)

### **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/)
* Local file system
* [Amazon S3](https://intellipaat.com/blog/what-is-amazon-s3/)
* [HBase](https://intellipaat.com/blog/what-is-apache-hbase/)
* [Cassandra](https://intellipaat.com/blog/what-is-apache-cassandra/), 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.

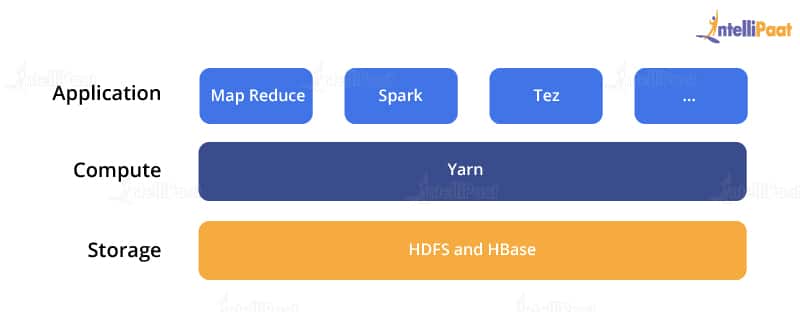
### **24. Roles of receivers in Apache Spark Streaming?**

Within [Apache Spark Streaming](https://intellipaat.com/blog/a-guide-to-apache-spark-streaming-tutorial/) 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/) 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/)**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/)***!***

### **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.

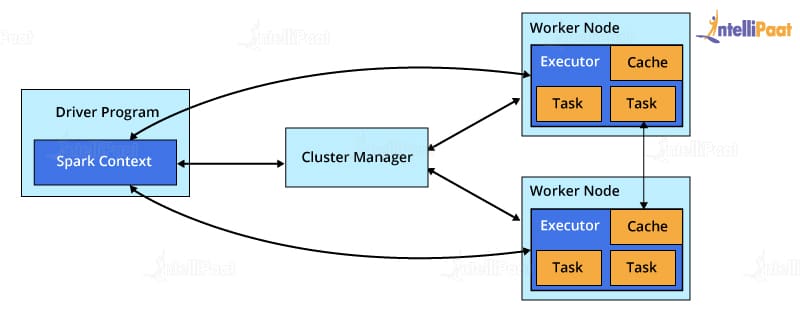
### **30. Name the types of Cluster Managers in Spark.**

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/). 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**

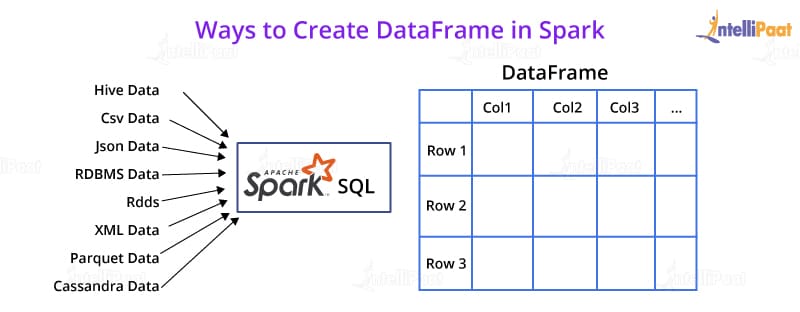
IntellipaatData = Array(2,4,6,8,10)

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/) 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.

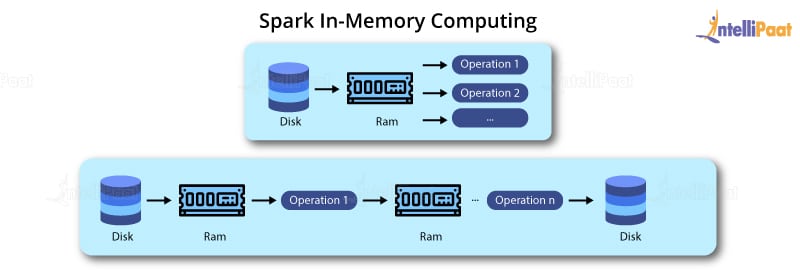
### **38. Which languages can Spark be integrated with?**

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.

***Interested in learning Spark? Take up our***[***Spark Training in Sydney***](https://intellipaat.com/apache-spark-scala-training-sydney/)***now!***

# Top Apache Spark Interview Questions You Should Prepare In 2023

Last updated on Nov 18,2022*296K Views*

**Share**

[image not found!](https://api.whatsapp.com/send?phone&text=Top%20Apache%20Spark%20Interview%20Questions%20You%20Should%20Prepare%20In%202023%20%20https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dwhatsapp%20%20%23Edureka&app_absent=0)[image not found!](https://www.linkedin.com/shareArticle?mini=true&url=https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dlinkedin&title=Top%20Apache%20Spark%20Interview%20Questions%20You%20Should%20Prepare%20In%202023&summary=Top%20Apache%20Spark%20Interview%20Questions%20You%20Should%20Prepare%20In%202023https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dlinkedin%23Edureka&source=https%3A%2F%2Fwww.edureka.co%2Fblog)[image not found!](https://twitter.com/intent/tweet?text=Top%20Apache%20Spark%20Interview%20Questions%20You%20Should%20Prepare%20In%202023%20%20%23Edureka&url=https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dtwitter&via=edurekain%20#Edureka)[image not found!](https://www.facebook.com/sharer/sharer.php?u=https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dfacebook&quote=Top%20Apache%20Spark%20Interview%20Questions%20You%20Should%20Prepare%20In%202023%20%20https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dfacebook%20%20%23Edureka)[image not found!](https://www.reddit.com/submit?url=https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dreddit&title=Top%20Apache%20Spark%20Interview%20Questions%20You%20Should%20Prepare%20In%202023)image not found!



[**Sandeep Dayananda**](https://www.edureka.co/blog/author/sandeepdedureka-co/)Sandeep Dayananda is a Research Analyst at Edureka. He has expertise in Big Data technologies like Hadoop & Spark, DevOps and Business Intelligence tools....

* [**1 Comments**](javascript:void(0))
* [**Bookmark**](https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/)

[image not found!](https://api.whatsapp.com/send?phone&text=Top%20Apache%20Spark%20Interview%20Questions%20You%20Should%20Prepare%20In%202023%20%20https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dwhatsapp%20%20%23Edureka&app_absent=0)[image not found!](https://www.linkedin.com/shareArticle?mini=true&url=https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dlinkedin&title=Top%20Apache%20Spark%20Interview%20Questions%20You%20Should%20Prepare%20In%202023&summary=Top%20Apache%20Spark%20Interview%20Questions%20You%20Should%20Prepare%20In%202023https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dlinkedin%23Edureka&source=https%3A%2F%2Fwww.edureka.co%2Fblog)[image not found!](https://twitter.com/intent/tweet?text=Top%20Apache%20Spark%20Interview%20Questions%20You%20Should%20Prepare%20In%202023%20%20%23Edureka&url=https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dtwitter&via=edurekain%20#Edureka)[image not found!](https://www.facebook.com/sharer/sharer.php?u=https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dfacebook&quote=Top%20Apache%20Spark%20Interview%20Questions%20You%20Should%20Prepare%20In%202023%20%20https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dfacebook%20%20%23Edureka)[image not found!](https://www.reddit.com/submit?url=https%3A%2F%2Fwww.edureka.co%2Fblog%2Finterview-questions%2Ftop-apache-spark-interview-questions-2016%3Futm_source%3Dsocialsharing%26utm_campaign%3Dreddit&title=Top%20Apache%20Spark%20Interview%20Questions%20You%20Should%20Prepare%20In%202023)image not found!

**1 / 1 Blog from Spark Interview Questions**

[](https://bit.ly/3Nrh5Qf)

2018 has been the year of Big Data – the year when big data and analytics made tremendous progress through innovative technologies, data-driven decision making and outcome-centric analytics. Worldwide revenues for big data and business analytics (BDA) will grow from $130.1 billion in 2016 to more than $203 billion in 2021 (source IDC). Prepare with these top **Apache Spark Interview Questions** to get an edge in the burgeoning Big Data market where global and local enterprises, big or small, are looking for a quality **Big Data and Hadoop experts.**

Want to Upskill yourself to get ahead in Career? Check out the [***Top Trending Technologies***](https://www.edureka.co/blog/top-10-trending-technologies/)Article***.***

As a big data professional, it is essential to know the right buzzwords, learn the right technologies and prepare the right answers to commonly asked Spark interview questions. With questions and answers around Spark Core, Spark Streaming, Spark SQL, GraphX, MLlib among others, this blog is your gateway to your next Spark job.

## ****Apache Spark Interview Questions And Answers****

### **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 |

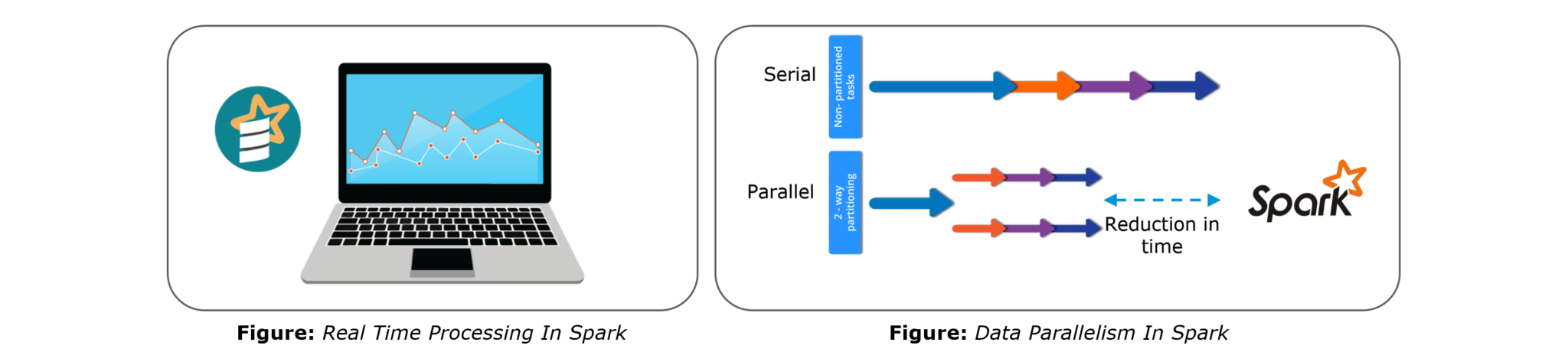
**Table:** Apache Spark versus Hadoop

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/) 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:

1. **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.
2. **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.
3. **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.
4. **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.
5. **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.
6. **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.
7. **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**=**&gt;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)](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)

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

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

[Explore Curriculum](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=curriculum" \t "_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 |

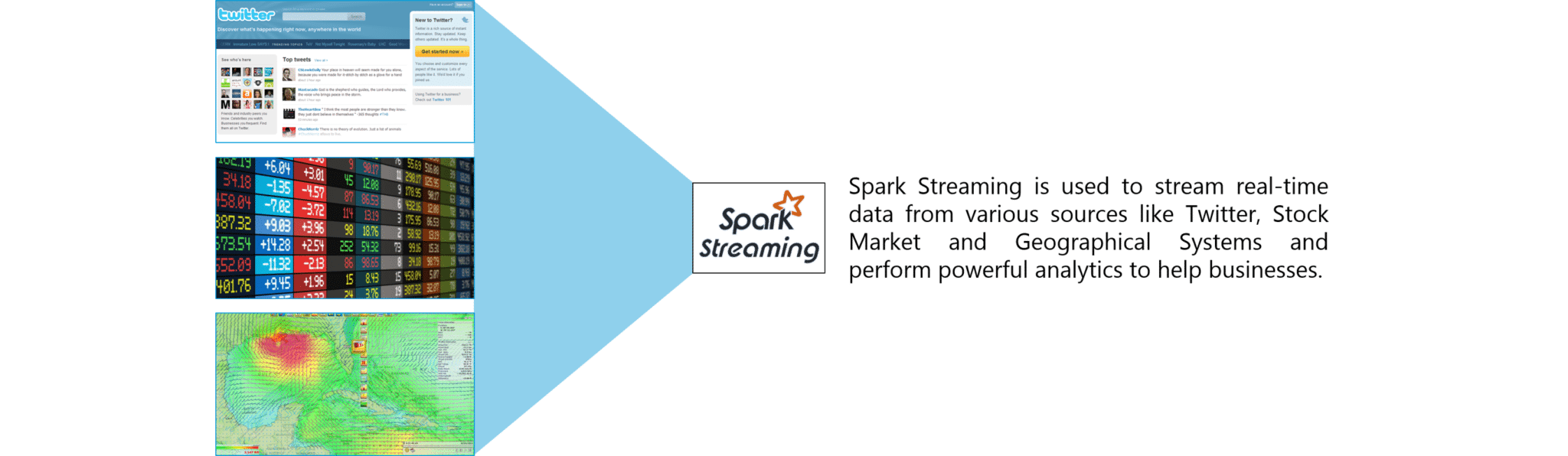
Special operations can be performed on RDDs in Spark using key/value pairs and such RDDs are referred to as Pair RDDs. Pair RDDs allow users to access each key in parallel. They have a reduceByKey() method that collects data based on each key and a join() method that combines different RDDs together,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

****

**Figure:** Spark Interview Questions – Spark Streaming

### **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.

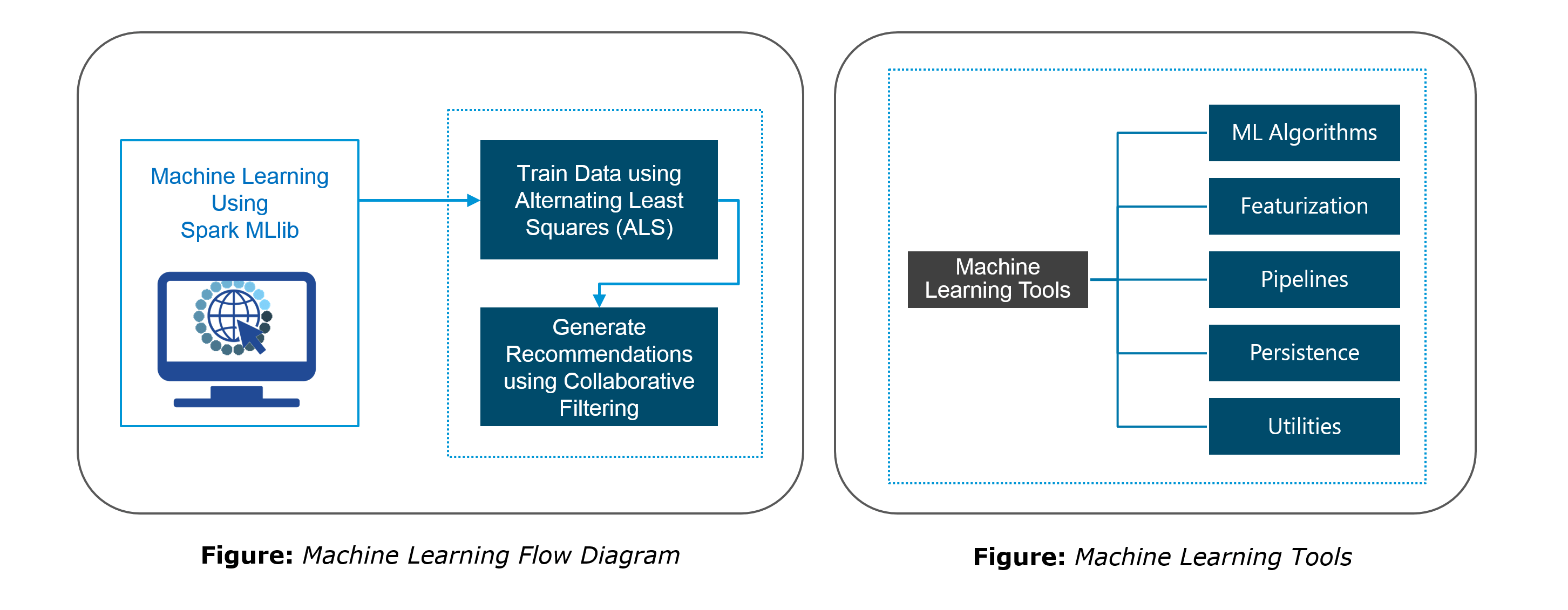
### **21. What is PageRank in GraphX?**

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

### Spark SQL - Spark Interview Questions - Edureka**24. What is a Parquet file?**

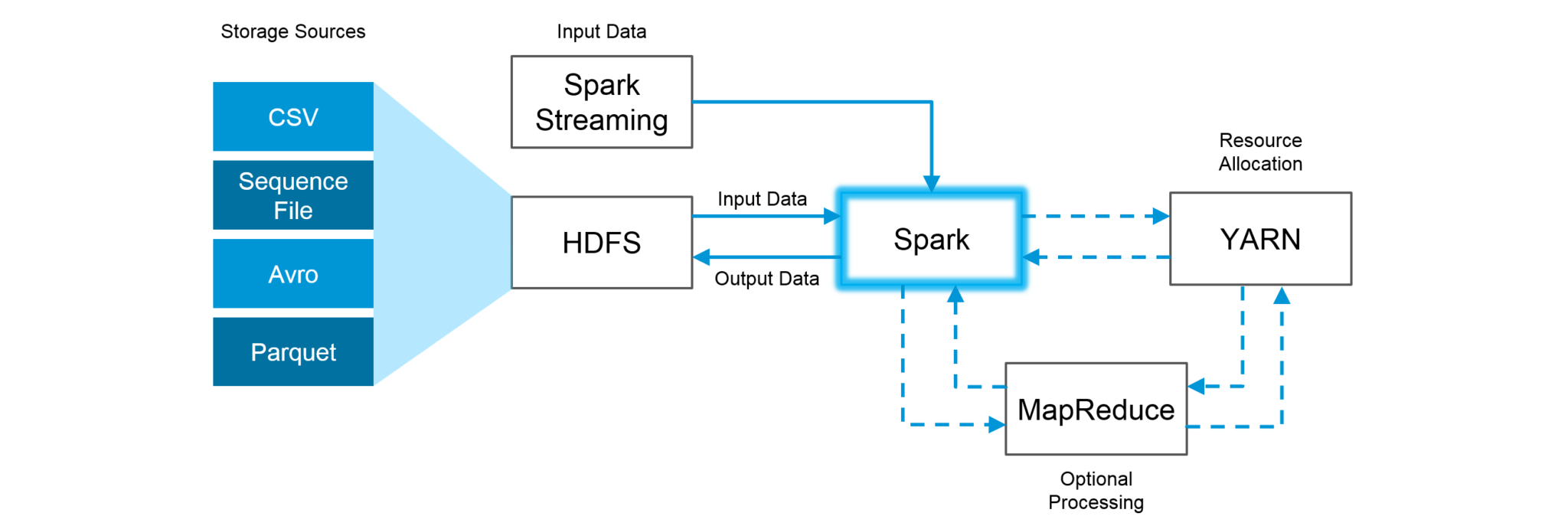
Parquet is a columnar format file supported by many other data processing systems. Spark SQL performs both read and write operations with Parquet file and consider it be one of the best big data analytics formats so far.

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.

**Figure:**Using Spark and Hadoop

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.

You can even check out the details of Big Data with the [Azure Data Engineer Associate](https://www.edureka.co/microsoft-azure-data-engineering-certification-course).

### **35. What is a Sparse Vector?**

A sparse vector has two parallel arrays; one for indices and the other for values. These vectors are used for storing non-zero entries to save space.

### Big Data Training

### [DP 203: DATA ENGINEERING ON MICROSOFT AZURE](https://www.edureka.co/microsoft-azure-data-engineering-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "DP 203: Data Engineering on Microsoft Azure" \t "_blank)

### [DP 203: Data Engineering on Microsoft Azure](https://www.edureka.co/microsoft-azure-data-engineering-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "DP 203: Data Engineering on Microsoft Azure" \t "_blank)

*[Reviews](https://www.edureka.co/microsoft-azure-data-engineering-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "DP 203: Data Engineering on Microsoft Azure" \t "_blank)*

**[5](https://www.edureka.co/microsoft-azure-data-engineering-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "DP 203: Data Engineering on Microsoft Azure" \t "_blank)**[(4838)](https://www.edureka.co/microsoft-azure-data-engineering-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "DP 203: Data Engineering on Microsoft Azure" \t "_blank)

### [BIG DATA HADOOP CERTIFICATION TRAINING COURSE](https://www.edureka.co/big-data-hadoop-training-certification?utm_source=blogbanner&utm_campaign=trending_course" \o "Big Data Hadoop Certification Training Course" \t "_blank)

### [Big Data Hadoop Certification Training Course](https://www.edureka.co/big-data-hadoop-training-certification?utm_source=blogbanner&utm_campaign=trending_course" \o "Big Data Hadoop Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/big-data-hadoop-training-certification?utm_source=blogbanner&utm_campaign=trending_course" \o "Big Data Hadoop Certification Training Course" \t "_blank)*

**[5](https://www.edureka.co/big-data-hadoop-training-certification?utm_source=blogbanner&utm_campaign=trending_course" \o "Big Data Hadoop Certification Training Course" \t "_blank)**[(170232)](https://www.edureka.co/big-data-hadoop-training-certification?utm_source=blogbanner&utm_campaign=trending_course" \o "Big Data Hadoop Certification Training Course" \t "_blank)

### [PYSPARK CERTIFICATION TRAINING COURSE](https://www.edureka.co/pyspark-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "PySpark Certification Training Course" \t "_blank)

### [PySpark Certification Training Course](https://www.edureka.co/pyspark-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "PySpark Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/pyspark-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "PySpark Certification Training Course" \t "_blank)*

**[5](https://www.edureka.co/pyspark-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "PySpark Certification Training Course" \t "_blank)**[(9391)](https://www.edureka.co/pyspark-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "PySpark Certification Training Course" \t "_blank)

### [LEVERAGING BIG DATA FOR BUSINESS INTELLIGENCE COURSE](https://www.edureka.co/leveraging-big-data-for-business-intelligence-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Leveraging Big Data for Business Intelligence Course" \t "_blank)

### [Leveraging Big Data for Business Intelligence Course](https://www.edureka.co/leveraging-big-data-for-business-intelligence-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Leveraging Big Data for Business Intelligence Course" \t "_blank)

*[Reviews](https://www.edureka.co/leveraging-big-data-for-business-intelligence-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Leveraging Big Data for Business Intelligence Course" \t "_blank)*

**[5](https://www.edureka.co/leveraging-big-data-for-business-intelligence-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Leveraging Big Data for Business Intelligence Course" \t "_blank)**[(2097)](https://www.edureka.co/leveraging-big-data-for-business-intelligence-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Leveraging Big Data for Business Intelligence Course" \t "_blank)

### [APACHE KAFKA CERTIFICATION TRAINING COURSE](https://www.edureka.co/kafka-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Apache Kafka Certification Training Course" \t "_blank)

### [Apache Kafka Certification Training Course](https://www.edureka.co/kafka-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Apache Kafka Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/kafka-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Apache Kafka Certification Training Course" \t "_blank)*

**[5](https://www.edureka.co/kafka-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Apache Kafka Certification Training Course" \t "_blank)**[(7766)](https://www.edureka.co/kafka-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Apache Kafka Certification Training Course" \t "_blank)

### [APACHE SPARK AND SCALA CERTIFICATION TRAINING COURSE](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Apache Spark and Scala Certification Training Course" \t "_blank)

### [Apache Spark and Scala Certification Training Course](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Apache Spark and Scala Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Apache Spark and Scala Certification Training Course" \t "_blank)*

**[5](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Apache Spark and Scala Certification Training Course" \t "_blank)**[(29244)](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Apache Spark and Scala Certification Training Course" \t "_blank)

### [SPLUNK CERTIFICATION TRAINING: POWER USER AND ADMIN](https://www.edureka.co/splunk-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Splunk Certification Training: Power User and Admin" \t "_blank)

### [Splunk Certification Training: Power User and Admin](https://www.edureka.co/splunk-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Splunk Certification Training: Power User and Admin" \t "_blank)

*[Reviews](https://www.edureka.co/splunk-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Splunk Certification Training: Power User and Admin" \t "_blank)*

**[5](https://www.edureka.co/splunk-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Splunk Certification Training: Power User and Admin" \t "_blank)**[(9164)](https://www.edureka.co/splunk-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Splunk Certification Training: Power User and Admin" \t "_blank)

### [ELK STACK TRAINING & CERTIFICATION](https://www.edureka.co/elk-stack-training-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "ELK Stack Training & Certification" \t "_blank)

### [ELK Stack Training & Certification](https://www.edureka.co/elk-stack-training-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "ELK Stack Training & Certification" \t "_blank)

*[Reviews](https://www.edureka.co/elk-stack-training-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "ELK Stack Training & Certification" \t "_blank)*

**[5](https://www.edureka.co/elk-stack-training-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "ELK Stack Training & Certification" \t "_blank)**[(2214)](https://www.edureka.co/elk-stack-training-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "ELK Stack Training & Certification" \t "_blank)

### [BIG DATA HADOOP ADMINISTRATION CERTIFICATION TRAINING](https://www.edureka.co/hadoop-administration-training-certification?utm_source=blogbanner&utm_campaign=trending_course" \o "Big Data Hadoop Administration Certification Training" \t "_blank)

### [Big Data Hadoop Administration Certification Training](https://www.edureka.co/hadoop-administration-training-certification?utm_source=blogbanner&utm_campaign=trending_course" \o "Big Data Hadoop Administration Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/hadoop-administration-training-certification?utm_source=blogbanner&utm_campaign=trending_course" \o "Big Data Hadoop Administration Certification Training" \t "_blank)*

**[5](https://www.edureka.co/hadoop-administration-training-certification?utm_source=blogbanner&utm_campaign=trending_course" \o "Big Data Hadoop Administration Certification Training" \t "_blank)**[(26040)](https://www.edureka.co/hadoop-administration-training-certification?utm_source=blogbanner&utm_campaign=trending_course" \o "Big Data Hadoop Administration Certification Training" \t "_blank)

Next

|  |  |
| --- | --- |
| 1 | Vectors.sparse(7,Array(0,1,2,3,4,5,6),Array(1650d,50000d,800d,3.0,3.0,2009,95054)) |

The above sparse vector can be used instead of dense vectors.

|  |  |
| --- | --- |
| 1 | **val** myHouse **=** Vectors.dense(4450d,2600000d,4000d,4.0,4.0,1978.0,95070d,1.0,1.0,1.0,0.0) |

### **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.

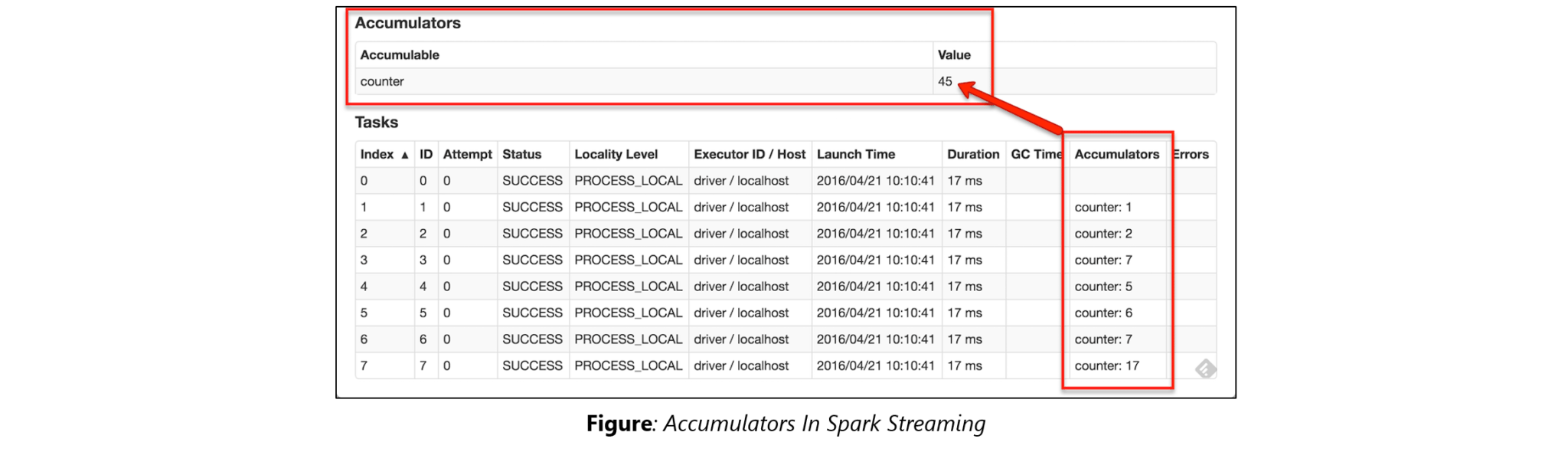
The most common way is to avoid operations ByKey, repartition or any other operations which trigger shuffles.

### **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.

### Broadcast Variables - Spark Interview Questions - Edureka**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.

### DStream Sliding Window - Spark Interview Questions - Edureka**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.

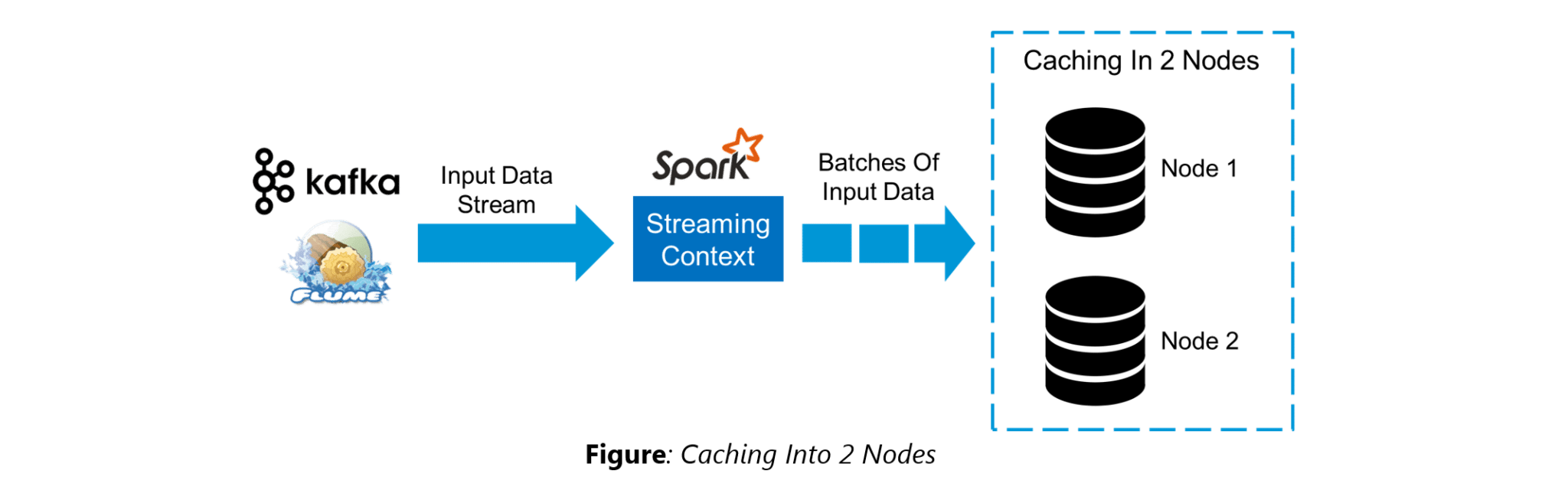
DStreams can be created from various sources like Apache Kafka, HDFS, and Apache Flume. DStreams have two operations:

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.

### DStream Filter - Spark Interview Questions - Edureka**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?**

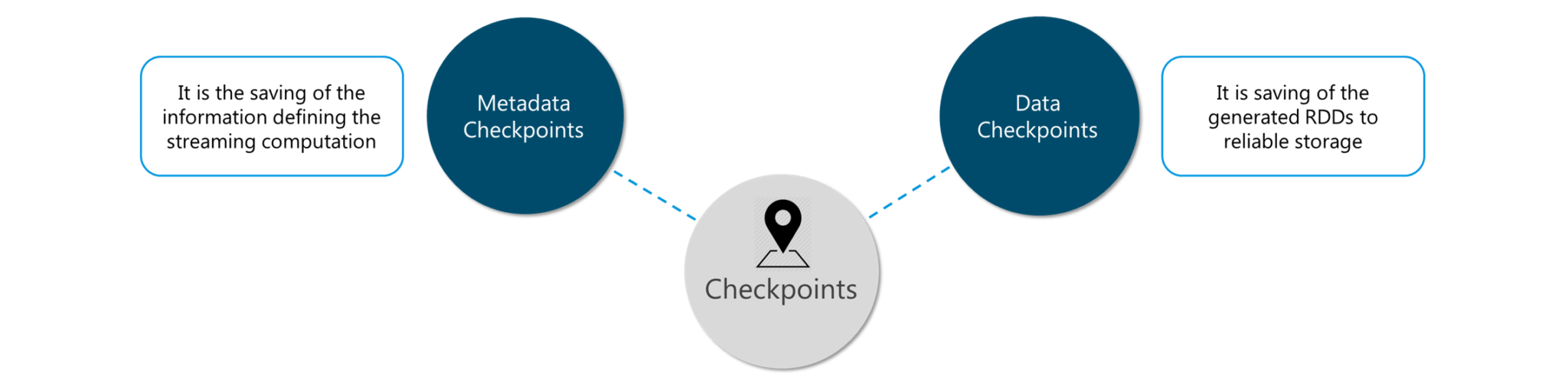
Apache Spark automatically persists the intermediary data from various shuffle operations, however, it is often suggested that users call persist () method on the RDD in case they plan to reuse it. Spark has various persistence levels to store the RDDs on disk or in memory or as a combination of both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

**Figure:** Spark Interview Questions – Checkpoints

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.

### Lazy Evaluation - Spark Interview Questions - Edureka**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.

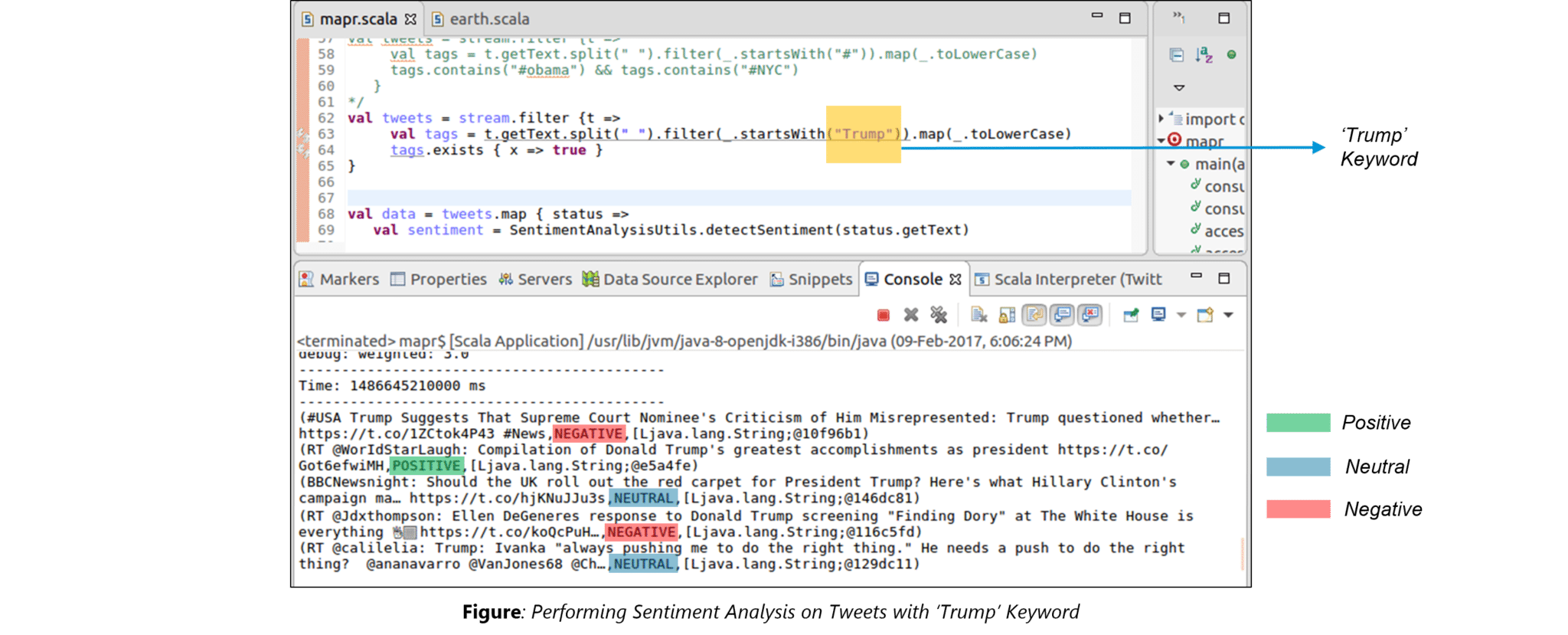
[[](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=batches)](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=batches" \t "_blank)

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

[Weekday / Weekend BatchesSee Batch Details](https://www.edureka.co/apache-spark-scala-certification-training?utm_source=blogbanner&utm_campaign=batches" \t "_blank)

Sentiment refers to the emotion behind a social media mention online. Sentiment Analysis is categorizing the tweets related to a particular topic and performing data mining using Sentiment Automation Analytics Tools.

Spark Streaming can be used to gather live tweets from around the world into the Spark program. This stream can be filtered using Spark SQL and then we can filter tweets based on the sentiment. The filtering logic will be implemented using MLlib where we can learn from the emotions of the public and change our filtering scale accordingly.



The above figure displays the sentiments for the tweets containing the word ‘Trump’.

Learn more about Spark Streaming in this tutorial: [Spark Streaming Tutorial | YouTube | Edureka](https://www.youtube.com/watch?v=uD_q4Rm4i2Q&list=PL9ooVrP1hQOGyFc60sExNX1qBWJyV5IMb)

I hope this set of Apache Spark interview questions will help you in preparing for your interview.

Further, I would recommend the following [Apache Spark Tutorial videos](https://www.youtube.com/playlist?list=PL9ooVrP1hQOGyFc60sExNX1qBWJyV5IMb) from Edureka to begin with.

## Top 10 Frequently Asked Apache Spark Interview Questions

1. [**What is Spark?**](https://mindmajix.com/apache-spark-interview-questions#spark)
2. [**What is Catchable?**](https://mindmajix.com/apache-spark-interview-questions#catchable)
3. [**What are Partitions?**](https://mindmajix.com/apache-spark-interview-questions#partitions)
4. [**How SparkSQL is different from HQL and SQL?**](https://mindmajix.com/apache-spark-interview-questions#different)
5. [**What is GraphX?**](https://mindmajix.com/apache-spark-interview-questions#graphx)
6. [**What is an RDD?**](https://mindmajix.com/apache-spark-interview-questions#rdd)
7. [**What is RDD Lineage?**](https://mindmajix.com/apache-spark-interview-questions#lineage)
8. [**What are Transformations?**](https://mindmajix.com/apache-spark-interview-questions#transformations)
9. [**How does the job support work?**](https://mindmajix.com/apache-spark-interview-questions#support)
10. [**Is there is a point in learning MapReduce, then?**](https://mindmajix.com/apache-spark-interview-questions#mapreduce)

## Apache Spark Interview Questions and Answer

### **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). |

### **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)***?*** |

### **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)](https://bit.ly/3if9dmk" \o "Subscribe MindMajix YouTube Channel" \t "_blank)

### **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)***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) 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) |

### **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) 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]

Facing technical problems in your current IT job, let us help you. MindMajix has highly technical people who can assist you in solving technical problems in your project.  
We have come across many developers in the USA, Australia, and other countries who have recently got the job but they are struggling to survive in the job because of less technical knowledge, exposure, and the kind of work given to them.  
We are here to help you.

Let us know your profile and the kind of help you are looking for and we shall do our best to help you out. The job support is provided by Mindmajix Technical experts who have more than 10 years of work experience in the IT technologies landscape.

### **46. How does the job support work?**

**Ans:**

* We see your project and technologies used, if we are 100% confident then we agree to support you.
* We work on the monthly basis
* No of hours of Support:  Based on customer need and the pricing also varies