SPARK Q&A

1)How does Spark relate to Apache Hadoop?

Spark is a fast and general processing engine compatible with Hadoop data. It can run in Hadoop clusters through YARN or Spark's standalone mode, and it can process data in HDFS, HBase, Cassandra, Hive, and any Hadoop InputFormat. It is designed to perform both batch processing (similar to MapReduce) and new workloads like streaming, interactive queries, and machine learning.

2) What is Apache Spark?

- Apache Spark 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)Compare Spark vs Hadoop MapReduce

Spark vs Hadoop

Criteria	Hadoop MapReduce	Apache Spark
Memory	Does not leverage the memory of the hadoop cluster to maximum.	Let's save data on memory with the use of RDD's.
Disk usage	MapReduce is disk oriented.	Spark caches data in-memory and ensures low latency.
Processing	Only batch processing is supported	Supports real-time processing through spark streaming.
Installation	Is bound to hadoop.	Is not bound to Hadoop.

Simplicity, Flexibility and Performance are the major advantages of using Spark over Hadoop.

- Spark is 100 times faster than Hadoop for big data processing as it stores the data in-memory, by placing it in Resilient Distributed Databases (RDD).
- Spark is easier to program as it comes with an interactive mode.
- It provides complete recovery using lineage graph whenever something goes wrong.

4) 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

5) What is 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.

6) What are the Multiple Formats supported in spark?

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.

7) What is 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.

8) What is Real Time Computation in Spark?

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.

9) What is 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.

10) How is Machine Learning performed in Spark?

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

11) 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

12)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.

13)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.

14)Name the components of Spark Ecosystem.

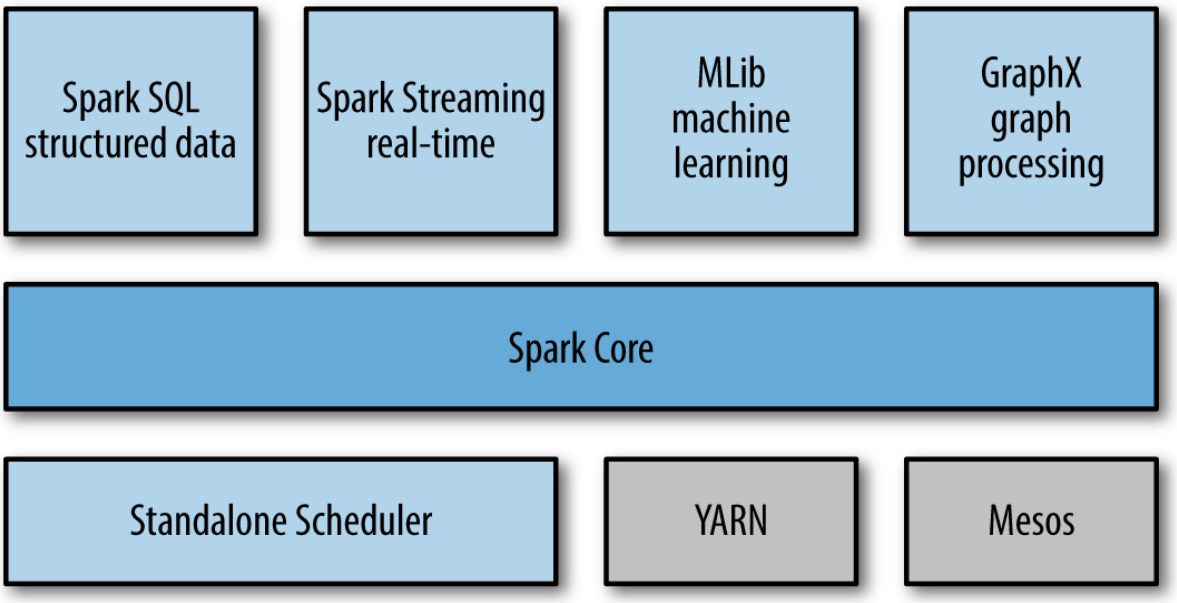
1.Spark Core: Base engine for large-scale parallel and distributed data processing

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



The Spark stack

15)what is Apache Spark Core?
Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

16)what is Spark SQL?
Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

17)what is Spark Streaming?
Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

18)what is MLlib (Machine Learning Library)?
MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of **Apache Mahout** (before Mahout gained a Spark interface).

19)what is GraphX?
GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

20) Explain about the different cluster managers in Apache Spark

The 3 different clusters managers supported in Apache Spark are:

- YARN
- Apache Mesos -Has rich resource scheduling capabilities and is well suited to run Spark along with other applications. It is advantageous when several users run interactive shells because it scales down the CPU allocation between commands.
- Standalone deployments – Well suited for new deployments which only run and are easy to set up.

21)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:

1Parallelized 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.

22)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' 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.

23)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.

24)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.

```
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.

25)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.

```
moviesData.saveAsTextFile("MoviesData.txt")
```

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

26)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

3Interacting with storage systems

27)What do you understand by Pair RDD?

Apache defines PairRDD functions class as

```
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.

28)What is Shark?

Most of the data users know only SQL and are not good at programming. Shark is a tool, developed for people who are from a database background - to access Scala MLib capabilities through Hive like SQL interface. Shark tool helps data users run Hive on Spark - offering compatibility with Hive metastore, queries and data.

29)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.

30)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.

31)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.

32)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

33)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.

34) 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.

35) 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.

36) 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.

37) 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.

38)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.

39) 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.

```
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.

```
val myHouse = Vectors.dense(4450d,2600000d,4000d,4.0,4.0,1978.0,95070d,1.0,1.0,1.0,0.0)
```

40)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.

41) 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.

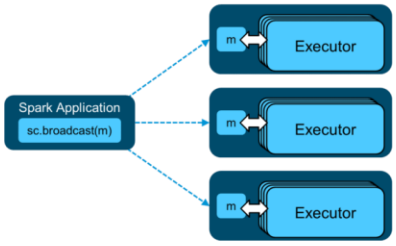


Figure: *Broadcasting A Value To Executors*

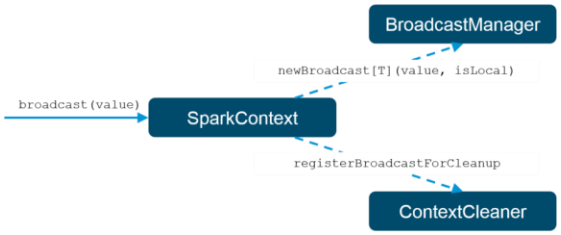
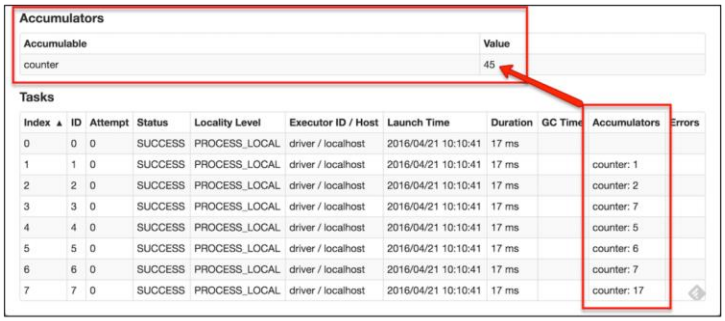


Figure: *SparkContext and Broadcasting*

42) 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.



Accumulators

Accumulable	Value
counter	45

Tasks

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

Figure: *Accumulators In Spark Streaming*

43) 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()*.

44) 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.

45) How can you remove the elements with a key present in any other RDD?

Use the subtractByKey () function

46)What is the difference between persist() and cache()

persist () allows the user to specify the storage level whereas cache () uses the default storage level

47)How can you achieve high availability in Apache Spark?

- Implementing single node recovery with local file system

- Using StandBy Masters with Apache ZooKeeper.

48) Hadoop uses replication to achieve fault tolerance. How is this achieved in Apache Spark?

Data storage model in Apache Spark is based on RDDs. RDDs help achieve fault tolerance through lineage. RDD always has the information on how to build from other datasets. If any partition of a RDD is lost due to failure, lineage helps build only that particular lost partition.

49) Explain about the core components of a distributed Spark application.

- Driver- The process that runs the main () method of the program to create RDDs and perform transformations and actions on them.
- Executor –The worker processes that run the individual tasks of a Spark job.
- Cluster Manager-A pluggable component in Spark, to launch Executors and Drivers. The cluster manager allows Spark to run on top of other external managers like Apache Mesos or YARN.

50)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 a 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.

51) What do you understand by SchemaRDD?

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

52) How can you launch Spark jobs inside Hadoop MapReduce?

Using SIMR (Spark in MapReduce) users can run any spark job inside MapReduce without requiring any admin rights.

53)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.

54)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.

55)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.

56)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.

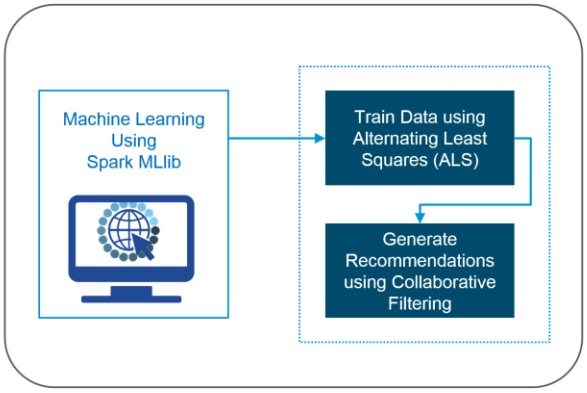


Figure: *Machine Learning Flow Diagram*

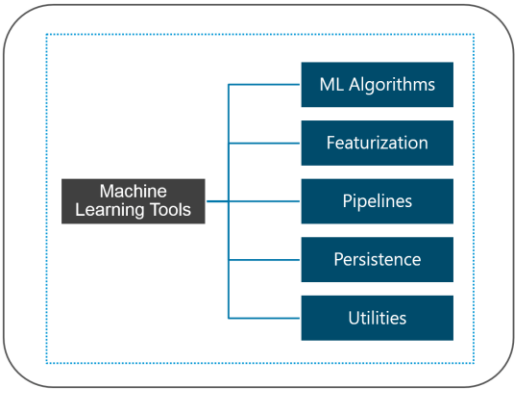


Figure: *Machine Learning Tools*

57)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

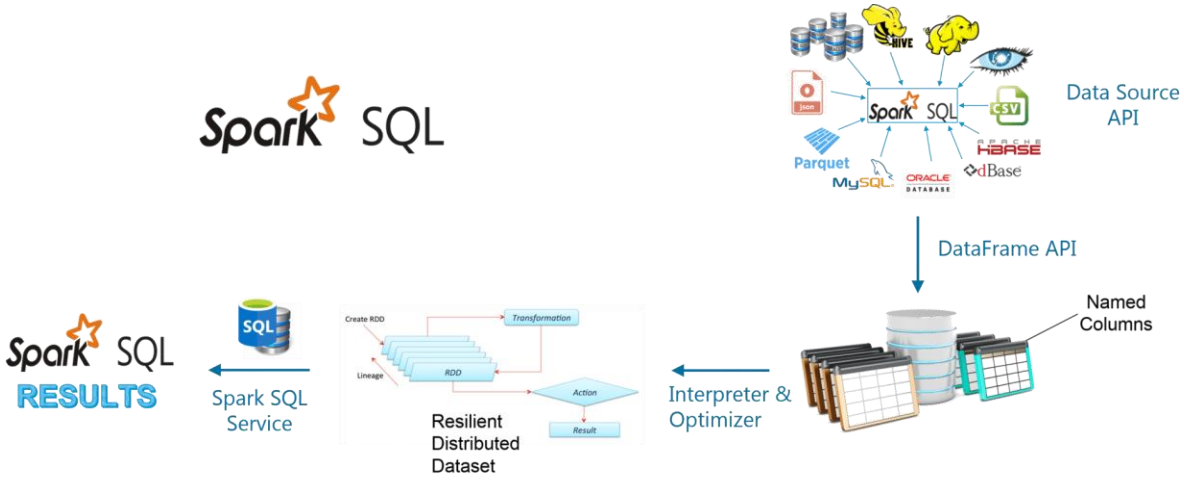


Figure: *The flow diagram represents a Spark SQL process using all the four libraries in sequence*

58)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.

59)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.

60)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*.

61)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.