Top 20 Apache Spark Interview Questions 2019

1. What is Apache Spark?

A. Apache Spark is a cluster computing framework which runs on a cluster of commodity hardware and performs data unification i.e., reading and writing of wide variety of data from multiple sources. In Spark, a task is an operation that can be a map task or a reduce task. Spark Context handles the execution of the job and also provides API's in different languages i.e., Scala, Java and Python to develop applications and faster execution as compared to MapReduce.

2. How is Spark different from MapReduce? Is Spark faster than MapReduce?

A. Yes, Spark is faster than MapReduce. There are few important reasons why Spark is faster than MapReduce and some of them are below:

- There is no tight coupling in Spark i.e., there is no mandatory rule that reduce must come after map.
- Spark tries to keep the data "in-memory" as much as possible.
  In MapReduce, the intermediate data will be stored in HDFS and hence takes longer time to get the data from a source but this is not the case with Spark.

3. Explain the Apache Spark Architecture. How to Run Spark applications?

- Apache Spark application contains two programs namely a Driver program and Workers program.
- A cluster manager will be there in-between to interact with these two cluster nodes. Spark Context will keep in touch with the worker nodes with the help of Cluster Manager.
- Spark Context is like a master and Spark workers are like slaves.
- Workers contain the executors to run the job. If any dependencies or arguments have to be passed then Spark Context will take care of that. RDD's will reside on the Spark Executors.
- You can also run Spark applications locally using a thread, and if you want to take advantage of distributed environments you can take the help of S3, HDFS or any other storage system.

4. What is RDD?

A. RDD stands for Resilient Distributed Datasets (RDDs). If you have large amount of data, and is not necessarily stored in a single system, all the data can be distributed across all the nodes and one subset of data is called as a partition which will be processed by a particular task. RDD's are very close to input splits in MapReduce.

More about RDD Here.

5. What is the role of coalesce () and repartition () in Map Reduce?

A. Both coalesce and repartition are used to modify the number of partitions in an RDD but Coalesce avoids full shuffle.

If you go from 1000 partitions to 100 partitions, there will not be a shuffle, instead each of the 100 new partitions will claim 10 of the current partitions and this does not require a shuffle.

Repartition performs a coalesce with shuffle. Repartition will result in the specified number of partitions with the data distributed using a hash practitioner.

6. How do you specify the number of partitions while creating an RDD? What are the functions?

A. You can specify the number of partitions while creating a RDD either by using the sc.textFile or by using parallelize functions as follows:

Val rdd = sc.parallelize(data,4)

val data = sc.textFile("path",4)

7. What are actions and transformations?

A. Transformations create new RDD's from existing RDD and these transformations are lazy and will not be executed until you call any action.

Eg: map(), filter(), flatMap(), etc.,

Actions will return results of an RDD.

Eg: reduce(), count(), collect(), etc.,

8. What is Lazy Evaluation?

A. If you create any RDD from an existing RDD that is called as transformation and unless you call an action your RDD will not be materialized the reason is Spark will delay the result until you really want the result because there could be some situations you have typed something and it went wrong and again you have to correct it in an interactive way it will increase the time and it will create un-necessary delays. Also, Spark optimizes the required calculations and takes intelligent decisions which is not possible with line by line code execution. Spark recovers from failures and slow workers.

9. Mention some Transformations and Actions

A. Transformations map (), filter(), flatMap()

Actions

reduce(), count(), collect()

10. What is the role of cache() and persist()?

A. Whenever you want to store a RDD into memory such that the RDD will be used multiple times or that RDD might have created after lots of complex processing in those situations, you can take the advantage of Cache or Persist.

You can make an RDD to be persisted using the persist() or cache() functions on it. The first time it is computed in an action, it will be kept in memory on the nodes.

When you call persist(), you can specify that you want to store the RDD on the disk or in the memory or both. If it is in-memory, whether it should be stored in serialized format or de-serialized format, you can define all those things.

cache() is like persist() function only, where the storage level is set to memory only.

11. What are Accumulators?

A. Accumulators are the write only variables which are initialized once and sent to the workers. These workers will update based on the logic written and sent back to the driver which will aggregate or process based on the logic.

Only driver can access the accumulator's value. For tasks, Accumulators are write-only. For example, it is used to count the number errors seen in RDD across workers.

12. What are Broadcast Variables?

A. Broadcast Variables are the read-only shared variables. Suppose, there is a set of data which may have to be used multiple times in the workers at different phases, we can share all those variables to the workers from the driver and every machine can read them.

13. What are the optimizations that developer can make while working with spark?

A. Spark is memory intensive, whatever you do it does in memory.

Firstly, you can adjust how long spark will wait before it times out on each of the phases of data locality (data local –> process local –> node local –> rack local –> Any).

Filter out data as early as possible. For caching, choose wisely from various storage levels.

Tune the number of partitions in spark.

14. What is Spark SQL?

A. Spark SQL is a module for structured data processing where we take advantage of SQL queries running on the datasets.

15. What is a Data Frame?

A. A data frame is like a table, it got some named columns which organized into columns. You can create a data frame from a file or from tables in hive, external databases SQL or NoSQL or existing RDD's. It is analogous to a table.

16. How can you connect Hive to Spark SQL?

A. The first important thing is that you have to place hive-site.xml file in conf directory of Spark. Then with the help of Spark session object we can construct a data frame as,
result = spark.sql("select * from <hive_table>")

17. What is GraphX?

A. Many times you have to process the data in the form of graphs, because you have to do some analysis on it. It tries to perform Graph computation in Spark in which data is present in files or in RDD's.

GraphX is built on the top of Spark core, so it has got all the capabilities of Apache Spark like fault tolerance, scaling and there are many inbuilt graph algorithms also. GraphX unifies ETL, exploratory analysis and iterative graph computation within a single system.

You can view the same data as both graphs and collections, transform and join graphs with RDD efficiently and write custom iterative algorithms using the pregel API.

GraphX competes on performance with the fastest graph systems while retaining Spark's flexibility, fault tolerance and ease of use.

18. What is PageRank Algorithm?

A. One of the algorithm in GraphX is PageRank algorithm. Pagerank measures the importance of each vertex in a graph assuming an edge from u to v represents an endorsements of v's importance by u.

For exmaple, in Twitter if a twitter user is followed by many other users, that particular will be ranked highly. GraphX comes with static and dynamic implementations of pageRank as methods on the pageRank object.

19. What is Spark Streaming?

A. Whenever there is data flowing continuously and you want to process the data as early as possible, in that case you can take the advantage of Spark Streaming. It is the API for stream processing of live data.

Data can flow for Kafka, Flume or from TCP sockets, Kenisis etc., and you can do complex processing on the data before you pushing them into their destinations. Destinations can be file systems or databases or any other dashboards.

20. What is Sliding Window?

A. In Spark Streaming, you have to specify the batch interval. For example, let's take your batch interval is 10 seconds, Now Spark will process the data whatever it gets in the last 10 seconds i.e., last batch interval time.

But with Sliding Window, you can specify how many last batches has to be processed. In the below screen shot, you can see that you can specify the batch interval and how many batches you want to process.

Apart from this, you can also specify when you want to process your last sliding window. For example you want to process the last 3 batches when there are 2 new batches. That is like when you want to slide and how many batches has to be processed in that window.