APACHE SPARK DEVELOPER INTERVIEW QUESTIONS SET

These interview questions are helpful for revising your basic concepts before appearing for Apache Spark developer position. This can be used by both interviewer and interviewee. However, it does not test the actual practical knowledge of the candidates. Its recommended that learner should use this just to revise their concepts and please read other materials as well as per the job position demands.

1. Why Spark, even Hadoop exists?

Answer: Below are few reasons.
Iterative Algorithm: Generally MapReduce is not good to process iterative algorithms like Machine Learning and Graph processing. Graph and Machine Learning algorithms are iterative by nature and less saves to disk, this type of algorithm needs data in memory to run algorithm steps again and again or less transfers over network means better performance.
In Memory Processing: MapReduce uses disk storage for storing processed intermediate data and also read from disks which is not good for fast processing. . Because Spark keeps data in Memory (Configurable), which saves lot of time, by not reading and writing data to disk as it happens in case of Hadoop.
Near real-time data processing: Spark also supports near real-time streaming workloads via Spark Streaming application framework.

2. Why both Spark and Hadoop needed?

Answer : Spark is often called cluster computing engine or simply execution engine. Spark uses many concepts from Hadoop MapReduce. Both Spark and Hadoop work together well. Spark with HDFS and YARN gives better performance and also simplifies the work distribution on cluster. As HDFSis storage engine for storing huge volume of data and Spark asa processing engine (In memory as well as more efficient data processing).
HDFS: It is used as a Storage engine for Spark as well as Hadoop.
YARN: It is a framework to manage Cluster using pluggable scedular.
Run other than MapReduce: With Spark you can run MapReduce algorithm as well asother higher level of operators for instance map(), filter(), reduceByKey(), groupByKey() etc.

3. How can you use Machine Learning library “SciKit library” which is written in Python, with Spark engine?

Answer: Machine learning tool written in Python, e.g. SciKit library, can be used as a Pipeline API in Spark MLlib or calling pipe().

4. Why Spark is good at low-latency iterative workloads e.g. Graphs and Machine Learning?

Answer: Machine Learning algorithms for instance logistic regression require many iterations before creating optimal resulting model. And similarly in graph algorithms which traverse all the nodes and edges. Any algorithm which needs many iteration before creating results can increase their performance when the intermediate partial results are stored in memory or at very fast solid state drives.

Spark can cache/store intermediate data in memory for faster model building and training. Also, when graph algorithms are processed then it traverses graphs one connection per iteration with the partial result in memory. Less disk access and network traffic can make a huge difference when you need to process lots of data.

5. Which all kind of data processing supported by Spark?

Answer: Spark offers three kinds of data processing using batch, interactive (Spark Shell), and stream processing with the unified API and data structures.

6. How do you define SparkContext?

Answer: It's an entry point for a Spark Job. Each Spark application starts by instantiating a Spark context. A Spark application is an instance of SparkContext. Or you can say, a Spark context constitutes a Spark application.

SparkContext represents the connection to a Spark execution environment (deployment mode).

A Spark contextcanbeusedtocreate RDDs, accumulatorsandbroadcastvariables, access Spark services and runjobs.

A“park context is essentially a client of “park's execution environment and it acts as the master of your Spark.

7. How canyou define SparkConf?

Answer: Spark properties control most application settings and are configured separately for each application. These properties can be set directly on a SparkConf passed to your SparkContext. SparkConf allows you to configure some of the common properties (e.g. master URL and application name), as well as arbitrary key-value pairs through the set() method. For example, we could initialize an application with two threads as follows:
Note that we run with local[2], meaning two threads - which represents “minimal” parallelism, which can help detect bugs that only exist when we run in a distributed context.

```
val conf = new SparkConf()

             .setMaster("local[2]")

             .setAppName("CountingSheep")

val sc = new SparkContext(conf)
```

8. Which all are the, ways to configure Spark Properties and order them least important to the most important.

Answer: There are the following ways to set up properties for Spark and user programs (in the order of importance from the least important to the most important):
conf/spark-defaults.conf - thedefault
--conf - the command line option used by spark-shell and spark-submit
SparkConf

9. What is the Default level of parallelism in Spark?

Answer: Default level of parallelism is the number of partitions when not specified explicitly by a user.

10. Is it possible to have multiple SparkContext in single JVM?

Answer: Yes, spark.driver.allowMultipleContexts is true (default: false). If true Spark logs warnings instead of throwing exceptions when multiple SparkContexts are active, i.e. multiple SparkContext are running in this JVM. When creating an instance of SparkContex.

11. Can RDD be shared between SparkContexts?

Answer: No, When an RDD is created; it belongs to and is completely owned by the Spark context it originated from. RDDs can't be shared between "parkContexts.

12. In Spark-Shell, which all contexts are available by default?

Answer: SparkContext and SQLContext

13. Give few examples , how RDD can be created using SparkContext

Answer: SparkContext allows you to create many different RDDs from input sources like:
"cala's collections: i.e. sc.parallelize(0 to 100)
Local or remote filesystems : sc.textFile("README.md")
Any Hadoop InputSource : using sc.newAPIHadoopFile

14. How would you brodcast, collection of values over the Sperk executors?

Answer: sc.broadcast("hello")

15. What is the advantage of broadcasting values across Spark Cluster?

Answer: Spark transfers the value to Spark executors once, and tasks can share it without incurring repetitive network transmissions when requested multiple times.

16. Can we broadcast an RDD?

Answer: Yes, you should not broadcast a RDD to use in tasks and Spark will warn you. It will not stop you, though.

17. How can we distribute JARs to workers?

Answer: The jar you specify with SparkContext.addJar will be copied to all the worker nodes.

18. How can you stop SparkContext and what is the impact if stopped?

Answer: You can stop a Spark context using SparkContext.stop() method. Stopping a Spark context stops the Spark Runtime Environment and effectively shuts down the entire Spark application.

19. Which scheduler is used by SparkContext by default?

Answer: By default, SparkContext uses DAGScheduler , but you can develop your own custom DAGScheduler implementation.

20. How would you the amount of memory to allocate to each executor?

Answer: SPARK_EXECUTOR_MEMORY sets the amount of memory to allocate to each executor.

21. How do you define RDD?

Answer: A Resilient Distributed Dataset (RDD), the basic abstraction in Spark. It represents an immutable, partitioned collection of elements that can be operated on in parallel. Resilient Distributed Datasets (RDDs) are a distributed memory abstraction that lets programmers perform in-memory computations on large clusters in a fault-tolerant manner.

Resilient: Fault-tolerant and so able torecomputed missing or damaged partitions on node failures with the help of RDD lineage graph.
Distributed: across clusters.
Dataset: is a collection of partitioned data.

22. What is Lazy evaluated RDD mean?

Answer: Lazy evaluated, i.e. the data inside RDD is not available or transformed until an action is executed that triggers the execution.

23. How would you control the number of partitions of a RDD?

Answer : You can control the number of partitions of a RDD using repartition or coalesce operations.

24. What are the possible operations on RDD

Answer: RDDs support two kinds ofoperations:
transformations - lazy operations that return another RDD.
actions - operations that trigger computation and return values.

25. How RDD helps parallel job processing?

Answer: Spark does jobs in parallel, and RDDs are split into partitions to be processed and written in parallel. Inside a partition, data is processed sequentially.

26. What is thetransformation?

Answer: A transformation is a lazy operation on a RDD that returns another RDD, likemap , flatMap
, filter , reduceByKey , join , cogroup , etc. Transformations are lazy and are not executed immediately, but only after an action have been executed.

27. How do you define actions?

Answer: An action is an operation that triggers execution of RDD transformations and returns a value (to a Spark driver - the user program). They trigger execution of RDD transformations to return values. Simply put, an action evaluates the RDD lineage graph.

You can think of actions as a valve and until no action is fired, the data to be processed is not even in the pipes, i.e. transformations. Only actions can materialize the entire processing pipeline with real data.

28. How can you create an RDD for a text file?

Answer: SparkContext.textFile

29. What is Preferred Locations

Answer: A preferred location (aka locality preferences or placement preferences) is a block location for an HDFS file where to compute each partition on.
def getPreferredLocations(split: Partition): Seq[String] specifies placement preferences for a partition in an RDD.

30. What is a RDD Lineage Graph

Answer: A RDD Lineage Graph (aka RDD operator graph) is a graph of the parent RDD of a RDD. It is built as a result of applying transformations to the RDD. A RDD lineage graph is hence a graph of what transformations need to be executed after an action has been called.

31. How execution starts and end on RDD or Spark Job

Answer: Execution Plan starts with the earliest RDDs (those with no dependencies on other RDDs or reference cached data) and ends with the RDD that produces the result of the action that has been called toexecute.

32. Give example of transformations that do trigger jobs

Answer: There area couple of transformations that do trigger jobs, e.g. sortBy , zipWithIndex , etc.

33. How many type of transformations exist?

Answer: There are two kinds of transformations:
narrow transformations
wide transformations

34. What is Narrow Transformations?

Answer: Narrow transformations are the result of map, filter and such that is from the data from a single partition only, i.e. it is self-sustained.
An output RDD has partitions with records that originate from a single partition in the parent RDD. Only a limited subset of partitions used to calculate the result. Spark groups narrow transformations as astage.

35. What is wide Transformations?

Answer: Wide transformations are the result of groupByKey and reduceByKey . The data required to compute the records in a single partition may reside in many partitions of the parent RDD.

All of the tuples with the same key must end up in the same partition, processed by the same task. To satisfy these operations, Spark must execute RDD shuffle, which transfers data across cluster and results in a new stage with a new set of partitions. (54)

36. Data is spread in all the nodes of cluster, how spark tries to process this data?

Answer: 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

37. How would you hint, minimum number of partitions while transformation ?

Answer: You can request forthe minimum number of partitions, usingthe second input parameter to many transformations.
scala> sc.parallelize(1 to 100, 2).count

Preferred way to set up the number of partitions for an RDD is to directly pass it as the second input parameter in the call like rdd = sc.textFile("hdfs://… /file.txt", 400) , where400 is the number of partitions. In this case, the partitioning makes for 400 splits that would be done by the Hadoop's TextInputFormat , not "park and it would work much faster. It'salso that the code spawns 400 concurrent tasks to try to load file.txt directly into 400 partitions.

38. How many concurrent task Spark can run for an RDD partition?

Answer: Spark can only run 1 concurrent task for every partition of an RDD, up to the number of cores in your cluster. So if you have a cluster with 50 cores, you want your RDDsto at least have 50 partitions (and probably 2-3x times that).

As far as choosing a "good" number of partitions, you generally want at least as many as the number of executors for parallelism. You can get this computed value by calling sc.defaultParallelism .

39. Which limits the maximum size of a partition?

Answer: The maximum size of a partition is ultimately limited by the available memory of an executor.

40. When Spark works with file.txt.gz, how many partitions can be created?

Answer: When using textFile with compressed files (file.txt.gz not file.txt or similar), Spark disables splitting that makes for an RDD with only 1 partition (as reads against gzipped files cannot be parallelized). In this case, to change the number of partitions you should do repartitioning.

Please note that Spark disables splitting for compressed files and creates RDDs with only 1 partition. In such cases, it's helpful to use sc.textFile('demo.gz') and do repartitioning using rdd.repartition(100) as follows:
rdd = sc.textFile('demo.gz') rdd =rdd.repartition(100)
With the lines, you end up with rdd to be exactly 100 partitions of roughly equal in size.

41. What is coalescetransformation?

Answer: The coalesce transformation is used to change the number of partitions. It can trigger RDD shuffling depending on the second shuffle boolean input parameter (defaults to false).

42. What is the difference between cache() and persist() method of RDD

Answer: RDDs can be cached (using RDD's cache() operation) or persisted (using RDD's persist(newLevel: StorageLevel) operation). The cache() operation is a synonym of persist() that uses the default storage level MEMORY_ONLY .

You have RDD storage level defined as MEMORY_ONLY_2 , what does _2 means ? Ans: number _2 in the name denotes 2 replicas

What is Shuffling?

Answer: Shuffling is a process of repartitioning (redistributing) data across partitions and may cause moving it across JVMs or even network when it is redistributed among executors.
Avoid shuffling at all cost. Think about ways to leverage existing partitions. Leverage partial aggregation to reduce data transfer.

Does shuffling change the number of partitions?

Answer: No, By default, shuffling doesn't change the number of partitions, but their content

What is the difference between groupByKey and use reduceByKey ?

Answer : Avoid groupByKey andusereduceByKeyor combineByKeyinstead. groupByKey shuffles all the data, which is slow.
reduceByKey shuffles only the results of sub-aggregations in each partition of the data.

Whenyoucall joinoperation on twopair RDDs e.g.(K, V) and (K, W), whatistheresult?
Answer: When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key [68]

What is checkpointing?
Answer: Checkpointing is a process of truncating RDD lineage graph and saving it to a reliable distributed (HDFS) or local file system. RDD checkpointing that saves the actual intermediate RDD data to a reliable distributed file system.

You mark an RDD for checkpointing by calling RDD.checkpoint() . The RDD will be saved to a file inside the checkpoint directory and all references to its parent RDDs will be removed. This function has to be called before any job has been executed on this RDD.

What do you mean by Dependencies in RDD lineage graph?
Answer: Dependency is a connection between RDDs after applying a transformation.

Which script will you use Spark Application, using spark-shell ?
Answer: You use spark-submit script to launch a Spark application, i.e. submit the application toa Spark deployment environment.

Define Spark architecture
Answer: Spark uses a master/worker architecture. There is a driver that talks to a single coordinator called master that manages workers in which executors run. The driver and the executors run in their own Java processes.

What is the purpose of Driver in Spark Architecture?
Answer :A Spark driver is the process that creates and owns an instance of SparkContext. It is your Spark application that launches the main method in which the instance of SparkContext is created.
Drive splits a Spark application into tasks and schedules them to run on executors.
A driver is where the task scheduler lives and spawns tasks across workers.
A driver coordinates workers and overall execution of tasks.

Can you define the purpose of master in Spark architecture?
Answer: A master is a running Spark instance that connects to a cluster manager for resources. The master acquires cluster nodes to run executors.

What are theworkers?
Answer: Workers or slaves are running Spark instances where executors live to execute tasks. They are the compute nodes in Spark. Aworker receives serialized/marshalled tasks that it runs in a thread pool.

Please explain, how worker's work, when a new Job submitted to them?
Answer: When SparkContext is created, each worker starts one executor. This is a separate java process or you can say new JVM, and it loads application jar in this JVM. Now executors connect back to your driver program and driver send them commands, like, foreach, filter, map etc. As soon as the driver quits, the executors shut down

Please define executors in detail?
Answer: Executors are distributed agents responsible for executing tasks. Executors provide in - memory storage for RDDs that are cached in Spark applications. When executors are started they register themselves with the driver and communicate directly to execute tasks. [112]

What is DAGSchedular and how it performs?
Answer: DAGScheduler is the scheduling layer of Apache Spark that implements stage-oriented scheduling, i.e. after an RDD action has been called it becomes a job that is then transformed into a set of stages that are submitted as TaskSets for execution.

DAGScheduler uses an event queue architecture in which a thread can post DAGSchedulerEvent events, e.g. a new job or stage being submitted, that DAGScheduler reads and executes sequentially.

What is stage, with regards to Spark Job execution?
Answer:Astageisaset of paralleltasks, oneperpartition of an RDD, that computepartialresultsof a function executed as part of a Spark job.

What is Task, with regards to Spark Job execution?
Answer: Task is an individual unit of work for executors to run. It is an individual unit of physical execution (computation) that runs on a single machine for parts of your Spark application on a data. All tasks in a stage should be completed before moving on to another stage.
A task can also be considered a computation in a stage on a partition in a given job attempt.
A Task belongs to a single stage and operates on a single partition (a part of an RDD).
Tasks are spawned one by one for each stage and data partition.

What is Speculative Execution of a tasks?
Answer: Speculative tasks ortask strugglers aretasks that runslower thanmost of the alltasks ina job.

Speculative execution of tasks is a health-check procedure that checks for tasks to be speculated, i.e. running slower in a stage than the median of all successfully completed tasks in a taskset . Such slow tasks will be re-launched in another worker. It will not stop the slow tasks, but run a new copy in parallel.

Which all cluster manager can be used with Spark?
Answer: Apache Mesos, Hadoop YARN, Spark standalone and
Spark local: Local node or on single JVM. Drivers and executor runs in same JVM. In this case same node will be used for execution.

What is a BlockManager?
Answer: Block Manager is a key-value store for blocks that acts as a cache. It runs on every node, i.e. a driver and executors, in a Spark runtime environment. It provides interfaces for putting and retrievingblocks bothlocallyand remotely into variousstores, i.e. memory, disk,and offheap.

A BlockManager manages the storage for most of the data in Spark, i.e. block that represent a cached RDD partition, intermediate shuffle data, and broadcast data.

What is Data locality / placement?
Answer: Spark relies on data locality or data placement or proximity to data source, that makes Spark jobssensitive towherethedataislocated. It istherefore important to haveSpark running on Hadoop YARN cluster if the data comes from HDFS.

With HDFS the Spark driver contacts NameNode about the DataNodes (ideally local) containing the various blocks of a file or directory as well as their locations (represented as InputSplits), and then schedules the work to the “parkWorkers. “park’s compute nodes / workers should be running on storagenodes.

What is master URL in local mode?
Answer: You can run Spark in local mode using local , local[n] or the most general local[*]. The URL says how many threads can be used in total:
local uses 1 thread only.
local[n] uses nthreads.
local[*] uses as many threads as the number of processors available to the Java virtual machine(it uses Runtime.getRuntime.availableProcessors() toknowthenumber).

Define components of YARN?
Answer: YARN components are below
ResourceManager: runs as a master daemon and manages ApplicationMasters and NodeManagers.
ApplicationMaster: is a lightweight process that coordinates the execution of tasks of an application and asks the ResourceManager for resource containers for tasks. It monitors tasks, restarts failed ones, etc. It can run any type of tasks, be them MapReduce tasks or Giraph tasks, or Spark tasks.
NodeManager offers resources (memory and CPU) as resource containers.
NameNode
Container: can run tasks, including ApplicationMasters.

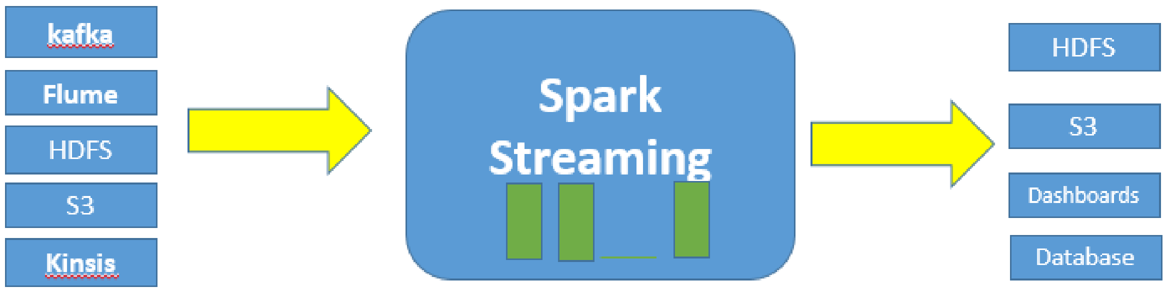
What is a Broadcast Variable?
Answer: Broadcast variables allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks.

How can you define Spark Accumulators?
Answer: This are similar to counters in Hadoop MapReduce framework, which gives information regarding completion of tasks, or how much data is processed etc.

What all are the data sources Spark can process?
Answer:
Hadoop File System(HDFS)
Cassandra (NoSQL databases)
HBase (NoSQL database)
S3 (Amazon WebService Storage : AWS Cloud)

What is Apache Parquet format?
Answer: Apache Parquet is a columnar storage format

What is Apache Spark Streaming?
Answer: Spark Streaming helps to process live stream data. Data can be ingested from many sources like Kafka, Flume, Twitter, ZeroMQ, Kinesis, or TCP sockets, andcanbe processed usingcomplex algorithms expressed with high-level functions like map, reduce, join and window.


kafka
Flume
HDFS
S3
Kinsis
Spark Streaming
HDFS
S3
Dashboards
Database