Apache Spark Interview Questions

Posted in Interview Questions (https://hackr.io/blog/category/interview-questions), Apache Spark (https://hackr.io/blog/category/apache-spark)

Akhil Bhadwal
(https://hackr.io/blog/author/akhilbhadwal)
Last Updated 05 Jan, 2021

Share:

(https://twitter.com/intent/tweet?text=+Apache+Spark+Interview+Questions+https%3A%2F%2Fhackr.io%2Fblog%2Fapache-spark-interview-questions) (https://www.linkedin.com/shareArticle?mini=true&url=https://hackr.io/blog/apache-spark-interview-questions) (http://www.reddit.com/submit?url=https://hackr.io/blog/apache-spark-interview-questions) (https://news.ycombinator.com/submitlink?u=https://hackr.io/blog/apache-spark-interview-questions) (https://api.whatsapp.com/send?text=https%3A%2F%2Fhackr.io%2Fblog%2Fapache-spark-interview-questions)

3 Comments

Table of Contents

Apache Spark is one of the most popular distributed, general-purpose cluster-computing frameworks. The open-source tool offers an interface for programming an entire computer cluster with implicit data parallelism and fault-tolerance features.

Top Apache Spark Interview Questions and Answers

Here we have compiled a list of the top Apache Spark interview questions. These will help you gauge your Apache Spark preparation for cracking that upcoming interview. Do you think you can get the answers right? Well, you'll only know once you've gone through it!

Question: Can you explain the key features of Apache Spark?

Answer:

- **Support for Several Programming Languages –** Spark code can be written in any of the four programming languages, namely Java, Python, R, and Scala (https://hackr.io/tutorials/learn-scala?ref=blog-post). It also provides high-level APIs in these programming languages. Additionally, Apache Spark provides shells in Python and Scala. The Python shell is accessed through the ./bin/pyspark directory, while for accessing the Scala shell one needs to go to the .bin/spark-shell directory.
- **Lazy Evaluation –** Apache Spark makes use of the concept of lazy evaluation (https://wiki.haskell.org/Lazy_evaluation), which is to delay the evaluation up until the point it becomes absolutely compulsory.
- **Machine Learning –** For big data processing, Apache Spark's MLib machine learning component is useful. It eliminates the need for using separate engines for processing and machine learning.
- **Multiple Format Support –** Apache Spark provides support for multiple data sources, including Cassandra, Hive, JSON, and Parquet. The Data Sources API offers a pluggable mechanism for accessing structured data via Spark SQL. These data sources can be much more than just simple pipes able to convert data and pulling the same into Spark.
- **Real-Time Computation –** Spark is designed especially for meeting massive scalability requirements. Thanks to its in-memory computation, Spark's computation is real-time and has less latency.
- **Speed –** For large-scale data processing, Spark can be up to 100 times faster than Hadoop MapReduce. Apache Spark is able to achieve this tremendous speed via controlled portioning. The distributed, general-purpose cluster-computing framework manages data by means of partitions that help in parallelizing distributed data processing with minimal network traffic.
- **Hadoop Integration –** Spark offers smooth connectivity with Hadoop. In addition to being a potential replacement for the Hadoop MapReduce functions, Spark is able to run on top of an extant Hadoop cluster by means of YARN for resource scheduling.

Question: What advantages does Spark offer over Hadoop MapReduce?

Answer:

- **Enhanced Speed –** MapReduce makes use of persistent storage for carrying out any of the data processing tasks. On the contrary, Spark uses in-memory processing that offers about 10 to 100 times faster processing than the Hadoop MapReduce.
- **Multitasking –** Hadoop only supports batch processing via inbuilt libraries. Apache Spark, on the other end, comes with built-in libraries for performing multiple tasks from the same core, including batch processing, interactive SQL queries, machine learning, and streaming.
- **No Disk-Dependency –** While Hadoop MapReduce is highly disk-dependent, Spark mostly uses caching and in-memory data storage.
- **Iterative Computation –** Performing computations several times on the same dataset is termed as iterative computation. Spark is capable of iterative computation while Hadoop MapReduce isn't.

Question: Please explain the concept of RDD (Resilient Distributed Dataset). Also, state how you can create RDDs in Apache Spark.

Answer: An RDD or Resilient Distribution Dataset is a fault-tolerant collection of operational elements that are capable to run in parallel. Any partitioned data in an RDD is distributed and immutable.

Fundamentally, RDDs are portions of data that are stored in the memory distributed over many nodes. These RDDs are lazily evaluated in Spark, which is the main factor contributing to the hastier speed achieved by Apache Spark. RDDs are of two types:

1. **Hadoop Datasets –** Perform functions on each file record in HDFS (Hadoop Distributed File System) or other types of storage systems
2. **Parallelized Collections –** Extant RDDs running parallel with one another

There are two ways of creating an RDD in Apache Spark:

- By parallelizing a collection in the Driver program. It makes use of SparkContext's parallelize() method. For instance:

```
method val DataArray = Array(22,24,46,81,101) val DataRDD = sc.parallelize(DataArray)
```

- By means of loading an external dataset from some external storage, including HBase, HDFS, and shared file system

Question: What are the various functions of Spark Core?

Answer: Spark Core acts as the base engine for large-scale parallel and distributed data processing. It is the distributed execution engine used in conjunction with the Java, Python, and Scala APIs that offer a platform for distributed ETL (Extract, Transform, Load) application development.

Various functions of Spark Core are:

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

Furthermore, additional libraries built on top of the Spark Core allow it to diverse workloads for machine learning, streaming, and SQL query processing.

Question: Please enumerate the various components of the Spark Ecosystem.

Answer:

1. GraphX – Implements graphs and graph-parallel computation
2. MLib – Used for machine learning
3. Spark Core – Base engine used for large-scale parallel and distributed data processing
4. Spark Streaming – Responsible for processing real-time streaming data
5. Spark SQL – Integrates Spark's functional programming (https://hackr.io/blog/functional-programming-concepts-advantages-disadvantages-applications) API with relational processing

Question: Is there any API available for implementing graphs in Spark?

Answer: GraphX is the API used for implementing graphs and graph-parallel computing in Apache Spark. It extends the Spark RDD with a Resilient Distributed Property Graph. It is a directed multi-graph that can have several edges in parallel.

Each edge and vertex of the Resilient Distributed Property Graph has user-defined properties associated with it. The parallel edges allow for multiple relationships between the same vertices.

In order to support graph computation, GraphX exposes a set of fundamental operators, such as joinVertices, mapReduceTriplets, and subgraph, and an optimized variant of the Pregel API.

The GraphX component also includes an increasing collection of graph algorithms and builders for simplifying graph analytics tasks.

Question: Tell us how will you implement SQL in Spark?

Answer: Spark SQL modules help in integrating relational processing with Spark's functional programming API. It supports querying data via SQL or HiveQL (Hive Query Language).

Also, Spark SQL supports a galore of data sources and allows for weaving SQL queries with code transformations. DataFrame API, Data Source API, Interpreter & Optimizer, and SQL Service are the four libraries contained by the Spark SQL.

Question: What do you understand by the Parquet file?

Answer: Parquet is a columnar format that is supported by several data processing systems. With it, Spark SQL performs both read as well as write operations. Having columnar storage has the following advantages:

- Able to fetch specific columns for access
- Consumes less space
- Follows type-specific encoding
- Limited I/O operations
- Offers better-summarized data

Question: Can you explain how you can use Apache Spark along with Hadoop?

Answer: Having compatibility with Hadoop is one of the leading advantages of Apache Spark. The duo makes up for a powerful tech pair. Using Apache Spark and Hadoop allows for making use of Spark's unparalleled processing power in line with the best of Hadoop's HDFS and YARN abilities.

Following are the ways of using Hadoop Components with Apache Spark:

- Batch & Real-Time Processing – MapReduce and Spark can be used together where the former handles the batch processing and the latter is responsible for real-time processing
- HDFS – Spark is able to run on top of the HDFS for leveraging the distributed replicated storage

- MapReduce – It is possible to use Apache Spark along with MapReduce in the same Hadoop cluster or independently as a processing framework
- YARN – Spark applications can run on YARN

Question: Name various types of Cluster Managers in Spark.

Answer:

1. Apache Mesos (https://en.wikipedia.org/wiki/Apache_Mesos) – Commonly used cluster manager
2. Standalone – A basic cluster manager for setting up a cluster
3. YARN – Used for resource management

Question: Is it possible to use Apache Spark for accessing and analyzing data stored in Cassandra databases?

Answer: Yes, it is possible to use Apache Spark for accessing as well as analyzing data stored in Cassandra (https://hackr.io/tutorials/learn-cassandra?ref=blog-post) databases using the Spark Cassandra Connector. It needs to be added to the Spark project during which a Spark executor talks to a local Cassandra node and will query only local data.

Connecting Cassandra with Apache Spark allows making queries faster by means of reducing the usage of the network for sending data between Spark executors and Cassandra nodes.

Question: What do you mean by the worker node?

Answer: Any node that is capable of running the code in a cluster can be said to be a worker node. The driver program needs to listen for incoming connections and then accept the same from its executors. Additionally, the driver program must be network addressable from the worker nodes.

A worker node is basically a slave node. The master node assigns work that the worker node then performs. Worker nodes process data stored on the node and report the resources to the master node. The master node schedule tasks based on resource availability.

Question: Please explain the sparse vector in Spark.

Answer: A sparse vector is used for storing non-zero entries for saving space. It has two parallel arrays:

1. One for indices
2. The other for values

An example of a sparse vector is as follows:

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

Question: How will you connect Apache Spark with Apache Mesos?

Answer: Step by step procedure for connecting Apache Spark with Apache Mesos is:

1. Configure the Spark driver program to connect with Apache Mesos
2. Put the Spark binary package in a location accessible by Mesos
3. Install Apache Spark in the same location as that of the Apache Mesos
4. Configure the spark.mesos.executor.home property for pointing to the location where the Apache Spark is installed

Question: Can you explain how to minimize data transfers while working with Spark?

Answer: Minimizing data transfers as well as avoiding shuffling helps in writing Spark programs capable of running reliably and fast. Several ways for minimizing data transfers while working with Apache Spark are:

- Avoiding – ByKey operations, repartition, and other operations responsible for triggering shuffles
- Using Accumulators – Accumulators provide a way for updating the values of variables while executing the same in parallel
- Using Broadcast Variables – A broadcast variable helps in enhancing the efficiency of joins between small and large RDDs

Question: What are broadcast variables in Apache Spark? Why do we need them?

Answer: Rather than shipping a copy of a variable with tasks, a broadcast variable helps in keeping a read-only cached version of the variable on each machine.

Broadcast variables are also used to provide every node with a copy of a large input dataset. Apache Spark tries to distribute broadcast variables by using effectual broadcast algorithms for reducing communication costs.

Using broadcast variables eradicates the need of shipping copies of a variable for each task. Hence, data can be processed quickly. Compared to an RDD lookup(), broadcast variables assist in storing a lookup table inside the memory that enhances retrieval efficiency.

Question: Please provide an explanation on DStream in Spark.

Answer: DStream is a contraction for Discretized Stream. It is the basic abstraction offered by Spark Streaming and is a continuous stream of data. DStream is received from either a processed data stream generated by transforming the input stream or directly from a data source.

A DStream is represented by a continuous series of RDDs, where each RDD contains data from a certain interval. An operation applied to a DStream is analogous to applying the same operation on the underlying RDDs. A DStream has two operations:

1. Output operations responsible for writing data to an external system
2. Transformations resulting in the production of a new DStream

It is possible to create DStream from various sources, including Apache Kafka, Apache Flume, and HDFS. Also, Spark Streaming provides support for several DStream transformations.

Question: Does Apache Spark provide checkpoints?

Answer: Yes, Apache Spark provides checkpoints. They allow for a program to run all around the clock in addition to making it resilient towards failures not related to application logic. Lineage graphs are used for recovering RDDs from a failure.

Apache Spark comes with an API for adding and managing checkpoints. The user then decides which data to the checkpoint. Checkpoints are preferred over lineage graphs when the latter are long and have wider dependencies.

Question: What are the different levels of persistence in Spark?

Answer: Although the intermediary data from different shuffle operations automatically persists in Spark, it is recommended to use the persist () method on the RDD if the data is to be reused.

Apache Spark features several persistence levels for storing the RDDs on disk, memory, or a combination of the two with distinct replication levels. These various persistence levels are:

- DISK_ONLY - Stores the RDD partitions only on the disk.
- MEMORY_AND_DISK - Stores RDD as deserialized Java objects in the JVM. In case the RDD isn't able to fit in the memory, additional partitions are stored on the disk. These are read from here each time the requirement arises.
- MEMORY_ONLY_SER - Stores RDD as serialized Java objects with one-byte array per partition.
- MEMORY_AND_DISK_SER - Identical to MEMORY_ONLY_SER with the exception of storing partitions not able to fit in the memory to the disk in place of recomputing them on the fly when required.
- MEMORY_ONLY - The default level, it stores the RDD as deserialized Java objects in the JVM. In case the RDD isn't able to fit in the memory available, some partitions won't be cached, resulting in recomputing

the same on the fly every time they are required.

- OFF_HEAP - Works like MEMORY_ONLY_SER but stores the data in off-heap memory.

Question: Can you list down the limitations of using Apache Spark?

Answer:

- It doesn't have a built-in file management system. Hence, it needs to be integrated with other platforms like Hadoop for benefitting from a file management system
- Higher latency but consequently, lower throughput
- No support for true real-time data stream processing. The live data stream is partitioned into batches in Apache Spark and after processing are again converted into batches. Hence, Spark Streaming is micro-batch processing and not truly real-time data processing
- Lesser number of algorithms available
- Spark streaming doesn't support record-based window criteria
- The work needs to be distributed over multiple clusters instead of running everything on a single node
- While using Apache Spark for cost-efficient processing of big data, its 'in-memory' ability becomes a bottleneck

Question: Define Apache Spark?

Answer: Apache Spark is an easy to use, highly flexible and fast processing framework which has an advanced engine that supports the cyclic data flow and in-memory computing process. It can run as a standalone in Cloud and Hadoop, providing access to varied data sources like Cassandra, HDFS, HBase, and various others.

Question: What is the main purpose of the Spark Engine?

Answer: The main purpose of the Spark Engine is to schedule, monitor, and distribute the data application along with the cluster.

Question: Define Partitions in Apache Spark?

Answer: Partitions in Apache Spark is meant to split the data in MapReduce by making it smaller, relevant, and more logical division of the data. It is a process that helps in deriving the logical units of data so that the speedy pace can be applied for data processing. Apache Spark is partitioned in Resilient Distribution Datasets (RDD).

Question: What are the main operations of RDD?

Answer: There are two main operations of RDD which includes:

1. Transformations
2. Actions

Question: Define Transformations in Spark?

Answer: Transformations are the functions that are applied to RDD that helps in creating another RDD. Transformation does not occur until action takes place. The examples of transformation are Map () and filer().

Question: What is the function of the Map ()?

Answer: The function of the Map () is to repeat over every line in the RDD and, after that, split them into new RDD.

Question: What is the function of filer()?

Answer: The function of filer() is to develop a new RDD by selecting the various elements from the existing RDD, which passes the function argument.

Question: What are the Actions in Spark?

Answer: Actions in Spark helps in bringing back the data from an RDD to the local machine. It includes various RDD operations that give out non-RDD values. The actions in Sparks include functions such as reduce() and take().

Question: What is the difference between reducing () and take() function?

Answer: Reduce() function is an action that is applied repeatedly until the one value is left in the last, while the take() function is an action that takes into consideration all the values from an RDD to the local node.

Question: What are the similarities and differences between coalesce () and repartition () in Map Reduce?

Answer: The similarity is that both Coalesce () and Repartition () in Map Reduce are used to modify the number of partitions in an RDD. The difference between them is that Coalesce () is a part of repartition(), which shuffles using Coalesce(). This helps repartition() to give results in a specific number of partitions with the whole data getting distributed by application of various kinds of hash practitioners.

Question: Define YARN in Spark?

Answer: YARN in Spark acts as a central resource management platform that helps in delivering scalable operations throughout the cluster and performs the function of a distributed container manager.

Question: Define PageRank in Spark? Give an example?

Answer: PageRank in Spark is an algorithm in Graphix which measures each vertex in the graph. For example, if a person on Facebook, Instagram, or any other social media platform has a huge number of followers than his/her page will be ranked higher.

Question: What is Sliding Window in Spark? Give an example?

Answer: A Sliding Window in Spark is used to specify each batch of Spark streaming that has to be processed. For example, you can specifically set the batch intervals and several batches that you want to process through Spark streaming.

Question: What are the benefits of Sliding Window operations?

Answer: Sliding Window operations have the following benefits:

- It helps in controlling the transfer of data packets between different computer networks.
- It combines the RDDs that falls within the particular window and operates upon it to create a new RDDs of the windowed DStream.
- It offers windowed computations to support the process of transformation of RDDs using the Spark Streaming Library.

Question: Define RDD Lineage?

Answer: RDD Lineage is a process of reconstructing the lost data partitions because Spark cannot support the data replication process in its memory. It helps in recalling the method used for building other datasets.

Question: What is a Spark Driver?

Answer: Spark Driver is referred to as the program which runs on the master node of the machine and helps in declaring the transformation and action on the data RDDs. It helps in creating SparkContext connected with the given Spark Master and delivers RDD graphs to Masters in the case where only the cluster manager runs.

Question: What kinds of file systems are supported by Spark?

Answer: Spark supports three kinds of file systems, which include the following:

- Amazon S3
- Hadoop Distributed File System (HDFS)
- Local File System.

Question: Define Spark Executor?

Answer: Spark Executor supports the SparkContext connecting with the cluster manager through nodes in the cluster. It runs the computation and data storing process on the worker node.

Question: Can we run Apache Spark on the Apache Mesos?

Answer: Yes, we can run Apache Spark on the Apache Mesos by using the hardware clusters that are managed by Mesos.

Question: Can we trigger automated clean-ups in Spark?

Answer: Yes, we can trigger automated clean-ups in Spark to handle the accumulated metadata. It can be done by setting the parameters, namely, “spark.cleaner.ttl.”

Question: What is another method than “Spark.cleaner.ttl” to trigger automated clean-ups in Spark?

Answer: Another method than “Spark.clener.ttl” to trigger automated clean-ups in Spark is by dividing the long-running jobs into different batches and writing the intermediary results on the disk.

Question: What is the role of Akka in Spark?

Answer: Akka in Spark helps in the scheduling process. It helps the workers and masters to send and receive messages for workers for tasks and master requests for registering.

Question: Define SchemaRDD in Apache Spark RDD?

Answer: SchemmaRDD is an RDD that carries various row objects such as wrappers around the basic string or integer arrays along with schema information about types of data in each column. It is now renamed as DataFrame API.

Question: Why is SchemaRDD designed?

Answer: SchemaRDD is designed to make it easier for the developers for code debugging and unit testing on the SparkSQL core module.

Question: What is the basic difference between Spark SQL, HQL, and SQL?

Answer: Spark SQL supports SQL and Hiver Query language without changing any syntax. We can join SQL and HQL table with the Spark SQL.

Conclusion

That completes the list of the 50 Top Spark interview questions. Going through these questions will allow you to check your Spark knowledge as well as help prepare for an upcoming Apache Spark interview.

You may want to check this best udemy course for performing better in Apache Spark interviews: Apache Hadoop Interview Questions Preparation Course (https://click.linksynergy.com/deeplink?id=jU79Zysihs4&mid=39197&murl=https://www.udemy.com/course/apache-hadoop-interview-questions-preparation-course/).

If you are looking for a fitting book for Apache interview questions, then buy this great book: 99 Apache Spark Interview Questions for Professionals: A GUIDE TO PREPARE FOR APACHE SPARK INTERVIEW QUESTIONS (https://geni.us/JNQ79kg).

How many of the aforementioned questions did you already know the answers to? Which questions should or shouldn’t have made it to the list? Let us know via comments! Consider checking out these best Spark tutorials (https://hackr.io/tutorials/learn-apache-spark?ref=blog-post) to further refine your Apache Spark skills.

All the best!

People are also reading:

- Python Interview Questions (https://hackr.io/blog/python-interview-questions)
- Best Machine Learning Books (https://hackr.io/blog/best-machine-learning-books)
- Data Analytics Certification (https://hackr.io/blog/best-data-analytics-certification)
- Machine Learning Certification (https://hackr.io/blog/machine-learning-certifications)
- Supervised vs Unsupervised Learning (https://hackr.io/blog/supervised-vs-unsupervised-learning)
- What is Machine Learning? (https://hackr.io/blog/what-is-machine-learning-definition-types)
- Top Data Science Tools (https://hackr.io/blog/data-science-tools)
- How to Become a Data Engineer? (https://hackr.io/blog/how-to-become-a-data-engineer)
- Hadoop vs Spark: Difference you should Know (https://hackr.io/blog/hadoop-vs-spark)
- Top Data Science Python Libraries (https://hackr.io/blog/top-data-science-python-libraries)

Interview Questions (https://hackr.io/blog/tag/interview-questions)

Spark Interview Questions (https://hackr.io/blog/tag/spark-interview-questions)

Apache Spark Interview Questions (https://hackr.io/blog/tag/apache-spark-interview-questions)

Apache Spark (https://hackr.io/blog/tag/apache-spark)

Share:

(https://twitter.com/intent/tweet?text=+Apache+Spark+Interview+Questions+https%3A%2F%2Fhackr.io%2Fblog%2Fapache-spark-interview-questions)

(https://www.linkedin.com/shareArticle?mini=true&url=https://hackr.io/blog/apache-spark-interview-questions)

(http://www.reddit.com/submit?url=https://hackr.io/blog/apache-spark-interview-questions)

(https://news.ycombinator.com/submitlink?u=https://hackr.io/blog/apache-spark-interview-questions)

(https://api.whatsapp.com/send?text=https%3A%2F%2Fhackr.io%2Fblog%2Fapache-spark-interview-questions)

Akhil Bhadwal

(https://hackr.io/blog/author/akhilbhadwal)

A Computer Science graduate interested in mixing up imagination and knowledge into enticing words. Been in the big bad world of content writing since 2014. In his free time, Akhil likes to play cards, do guitar jam, and write weird fiction. View all posts by the Author (https://hackr.io/blog/author/akhilbhadwal)

Related Posts

(https://hackr.io/blog/sql-server-interview-questions)

50 Best SQL Server Interview Questions and Answers

(https://hackr.io/blog/sql-server-interview-questions) **Read More (https://hackr.io/blog/sql-server-interview-questions)**

(https://hackr.io/blog/software-testing-interview-questions)

Software Testing Interview Questions and Answers

(https://hackr.io/blog/software-testing-interview-questions)

Read More (https://hackr.io/blog/software-testing-interview-questions)

(https://hackr.io/blog/css-interview-questions)

40+ Top CSS Interview Questions and Answers

(https://hackr.io/blog/css-interview-questions) **Read More (https://hackr.io/blog/css-interview-questions)**

Leave a comment

Email address*

Enter email

Your email will not be published

Name*

Name

Comment*

SUBMIT

Micheal Page
Is spark a programming language?

Reply

Karla Wallace
Spark is very much a programming language. It's originally based on the Ada computer programming language.

Reply

Louise Green
Why do people use Spark?

Reply

Eddie Price
Spark is extensively used in the field of Data Science. Data scientists across the globe use Spark around the cases which involve investigative and operational analytics.

Reply

Brent Woods
Which type of processing Apache Spark can handle?

Reply

Ken Cole
The best processing for Apache Spark is Java, Scala, or Python. Use these three and you will witness the best of Apache Spark.

Reply

Related Tutorials

 Apache Spark

(https://hackr.io/tutorials/learn-apache-spark)

 Hadoop

(https://hackr.io/tutorials/learn-hadoop-big-data)

 D3.js

(https://hackr.io/tutorials/learn-d3-js)

 Apache Kafka

(https://hackr.io/tutorials/learn-apache-kafka)

 RabbitMQ

(https://hackr.io/tutorials/learn-rabbitmq)

Recommended Learning

Apache Spark in Python: Beginner's Guide (www.datacamp.com)

(https://hackr.io/tutorial/apache-spark-in-python-beginners-guide)

Taming Big Data with Apache Spark and Python (www.udemy.com)

(https://hackr.io/tutorial/taming-big-data-with-apache-spark-and-python)

Apache Spark SQL (www.experfy.com)

(https://hackr.io/tutorial/apache-spark-sql)

VIEW MORE (https://hackr.io/tutorials/learn-apache-spark)

Disclosure: This page may contain affliate links, meaning when you click the links and make a purchase, we receive a commission.