PYSPARK - THEORY Q&A

BY - SHUBHAM WADEKAR

1. What is Apache Spark?

Apache Spark is an open-source, distributed computing system that provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. It is designed to process large-scale data efficiently.

2. Why Apache Spark?

Apache Spark is used because it is faster than traditional big data tools like Hadoop MapReduce due to its in-memory processing capabilities, supports multiple languages (Scala, Python, R, Java), provides libraries for various tasks (SQL, machine learning, graph processing, etc.), and has robust fault tolerance.

3. What are the components of the Apache Spark Ecosystem?

The main components are:

- **Spark Core**: The foundational engine for large-scale parallel and distributed data processing.
- **Spark SQL**: For structured data processing.
- **Spark Streaming**: For real-time data processing.
- **MLlib**: A library for scalable machine learning.
- **GraphX**: For graph and graph-parallel computation.

4. What is Spark Core?

Spark Core is the general execution engine for the Spark platform, responsible for tasks such as scheduling, distributing, and monitoring applications.

5. Which languages does Apache Spark support?

Apache Spark supports:

- Scala
- Python
- Java
- R
- SQL

6. How is Apache Spark better than Hadoop?

Spark is better in several ways, including faster processing due to in-memory computation, ease of use with APIs for various programming languages, flexibility with built-in libraries for diverse tasks, and a rich set of APIs for transformations and actions.

7. What are the different methods to run Spark over Apache Hadoop?

Spark can run on Hadoop in the following modes:

- **Standalone**: Using its cluster manager.
- **YARN**: Hadoop’s cluster manager.
- **Mesos**: Another cluster manager.

8. What is SparkContext in Apache Spark?

`SparkContext` is the entry point for any Spark application. It acts as a connection to the Spark cluster, allowing Spark jobs to be executed.

9. What is SparkSession in Apache Spark?

`SparkSession` is the unified entry point to work with DataFrames, Datasets, and SQL in Apache Spark. It replaces `SQLContext` and `HiveContext`.

10. SparkSession vs. SparkContext in Apache Spark

`SparkSession` is a combination of `SQLContext`, `HiveContext`, and `SparkContext` to provide a single point of entry to interact with Spark.

11. What are the abstractions of Apache Spark?

The primary abstractions are:

- **RDD (Resilient Distributed Dataset)**
- **DataFrames**
- **Datasets**

12. How can we create RDD in Apache Spark?

RDDs can be created in three ways:

- **Parallelizing a collection** in your program.
- **Referencing a dataset** in an external storage system (e.g., HDFS, S3, etc.).
- **Transforming** an existing RDD.

13. Why is Spark RDD immutable?

RDDs are immutable to provide fault tolerance and support functional programming principles, allowing Spark to rebuild lost data from the lineage information.

14. Explain the term paired RDD in Apache Spark.

Paired RDDs are RDDs where each element is a pair (key-value). They are used for operations like aggregation, grouping, and joins.

15. How is RDD in Spark different from Distributed Storage Management?

RDD is an in-memory data structure optimized for processing, while Distributed Storage (like HDFS) focuses on data storage and retrieval.

16. Explain transformation and action in RDD in Apache Spark.

- **Transformation**: Lazy operations that define a new RDD without executing until an action is called (e.g., `map`, `filter`).
- **Action**: Triggers the execution of transformations (e.g., `count`, `collect`).

17. What are the types of Apache Spark transformations?

Transformations can be narrow (e.g., `map`, `filter`) or wide (e.g., `groupByKey`, `reduceByKey`).

18. Explain the RDD properties.

RDD properties include:

- **Immutability**: Once created, RDDs cannot be changed.
- **Partitioned**: Distributed across various nodes in the cluster.
- **Lazy evaluation**: Operations are computed when an action is called.
- **Fault tolerance**: Recomputed using lineage information.

19. What is a lineage graph in Apache Spark?

A lineage graph tracks the sequence of transformations that created an RDD, used for recomputing lost data due to node failures.

20. Explain the terms Spark Partitions and Partitioners.

- **Partitions**: Logical division of data in RDDs, physically stored across nodes.
- **Partitioner**: Determines how data is distributed across partitions (e.g., `HashPartitioner`, `RangePartitioner`).

21. By default, how many partitions are created in RDD in Apache Spark?

By default, Spark creates partitions based on the number of cores available or the input file's HDFS block size.

22. What is Spark DataFrames?

DataFrames are distributed collections of data organized into named columns, similar to tables in a relational database.

23. What are the benefits of DataFrame in Spark?

Benefits include optimizations (Catalyst query optimizer), improved performance, and easier manipulation using SQL-like syntax.

24. What is Spark Dataset?

A Dataset is a distributed collection of data that provides type safety and object-oriented programming interfaces.

25. What are the advantages of datasets in Spark?

Advantages include compile-time type safety, optimizations through Tungsten, and the ability to leverage JVM object serialization.

26. What is Directed Acyclic Graph (DAG) in Apache Spark?

A DAG in Spark represents a sequence of computations performed on data, where each node is an RDD and edges represent transformations. It's used to optimize execution plans.

27. What is the need for Spark DAG?

The DAG allows Spark to optimize execution by scheduling tasks efficiently, minimizing data shuffling, and managing dependencies.

28. What is the difference between DAG and Lineage?

- **DAG**: Represents the entire execution plan of a Spark application.
- **Lineage**: Tracks transformations on a particular RDD, useful for fault recovery.

29. What is the difference between Caching and Persistence in Apache Spark?

- **Caching**: Default storage level is in-memory (`MEMORY_ONLY`).
- **Persistence**: Allows choosing different storage levels (disk, memory, etc.) for storing RDDs.

30. What are the limitations of Apache Spark?

Limitations include high memory consumption, limited built-in libraries compared to Hadoop, not suitable for small data or real-time streaming without specific tools.

31. Different Running Modes of Apache Spark

Spark can run in:

- **Local mode**: Single machine.
- **Standalone mode**: Using its cluster manager.
- **YARN mode**: On Hadoop's cluster manager.
- **Mesos mode**: On Mesos cluster manager.
- **Kubernetes mode**: On Kubernetes.

32. What are the different ways of representing data in Spark?

Data can be represented as:

- **RDDs (Resilient Distributed Datasets)**

- **DataFrames**
- **Datasets**

33. What is Write-Ahead Log (WAL) in Spark?

Write-Ahead Log is a fault-tolerance mechanism where every received data is first written to a log file (disk) before processing, ensuring no data loss.

34. Explain Catalyst Query Optimizer in Apache Spark.

Catalyst is Spark SQL's query optimizer that uses rule-based and cost-based optimization techniques to generate efficient execution plans.

35. What are shared variables in Apache Spark?

Shared variables are variables that can be used by tasks running on different nodes:

- **Broadcast variables**: Efficiently share read-only data across nodes.
- **Accumulators**: Used for aggregating information (e.g., sums) across tasks.

36. How does Apache Spark handle accumulated metadata?

Spark stores metadata like lineage information, partition data, and task details in the driver and worker nodes, managing it using its DAG scheduler.

37. What is Apache Spark's Machine Learning Library?

MLlib is Spark's scalable machine learning library, which provides algorithms and utilities for classification, regression, clustering, collaborative filtering, and more.

38. List commonly used Machine Learning Algorithms.

Common algorithms in Spark MLlib include:

- Linear Regression
- Logistic Regression
- Decision Trees
- Random Forests
- Gradient-Boosted Trees
- K-Means Clustering

39. What is the difference between DSM and RDD?

- **DSM (Distributed Storage Management)**: Focuses on data storage across clusters.
- **RDD (Resilient Distributed Dataset)**: Focuses on distributed data processing with fault tolerance.

40. List the advantage of Parquet file in Apache Spark.

Advantages of Parquet files:

- Columnar storage format, optimized for read-heavy workloads.
- Efficient compression and encoding schemes.
- Schema evolution support.

41. What is lazy evaluation in Spark?

Lazy evaluation defers execution until an action is performed, optimizing the execution plan by reducing redundant computations.

42. What are the benefits of Spark lazy evaluation?

Benefits include:

- Reducing the number of passes over data.
- Optimizing the computation process.
- Decreasing execution time.

43. How much faster is Apache Spark than Hadoop?

Apache Spark is generally up to 100x faster than Hadoop for in-memory processing and up to 10x faster for on-disk data.

44. What are the ways to launch Apache Spark over YARN?

Spark can be launched over YARN in:

- **Client mode**: Driver runs on the client machine.
- **Cluster mode**: Driver runs inside YARN cluster.

45. Explain various cluster managers in Apache Spark.

Spark supports:

- **Standalone Cluster Manager**: Default cluster manager.
- **Apache Mesos**: A general-purpose cluster manager.
- **Hadoop YARN**: A resource manager for Hadoop clusters.
- **Kubernetes**: For container orchestration.

46. What is Speculative Execution in Apache Spark?

Speculative execution is a mechanism to detect slow-running tasks and run duplicates on other nodes to speed up the process.

47. How can data transfer be minimized when working with Apache Spark?

Data transfer can be minimized by:

- Reducing shuffling and repartitioning.
- Using broadcast variables.
- Efficient data partitioning.

48. What are the cases where Apache Spark surpasses Hadoop?

Apache Spark outperforms Hadoop in scenarios involving iterative algorithms, in-memory computations, real-time analytics, and complex data processing workflows.

49. What is an action, and how does it process data in Apache Spark?

An action is an operation that triggers the execution of transformations (e.g., `count`, `collect`), performing computations and returning a result.

50. How is fault tolerance achieved in Apache Spark?

Fault tolerance is achieved through lineage information, allowing RDDs to be recomputed from scratch if a partition is lost.

51. What is the role of the Spark Driver in Spark applications?

The Spark Driver is responsible for converting the user's code into tasks, scheduling them on executors, and collecting the results.

52. What is a worker node in an Apache Spark cluster?

A worker node is a machine in a Spark cluster where the actual data processing tasks are executed.

53. Why is Transformation lazy in Spark?

Transformations are lazy to build an optimized execution plan (DAG) and to avoid unnecessary computation.

54. Can I run Apache Spark without Hadoop?

Yes, Spark can run independently using its built-in cluster manager or other managers like Mesos and Kubernetes.

55. Explain Accumulator in Spark.

An accumulator is a variable used for aggregating information across executors, like counters in MapReduce.

56. What is the role of the Driver program in a Spark Application?

The Driver program coordinates the execution of tasks, maintains the SparkContext, and communicates with the cluster manager.

57. How to identify that a given operation is a Transformation or Action in your program?

Transformations return RDDs (e.g., `map`, `filter`), while actions return non-RDD values (e.g., `collect`, `count`).

58. Name the two types of shared variables available in Apache Spark.

- **Broadcast Variables**
- **Accumulators**

59. What are the common faults of developers while using Apache Spark?

Common faults include:

- Inefficient data partitioning.
- Excessive shuffling and data movement.
- Inappropriate use of transformations and actions.
- Not leveraging caching and persistence properly.

60. By Default, how many partitions are created in RDD in Apache Spark?

The default number of partitions is based on the number of cores available in the cluster or the HDFS block size.

61. Why do we need compression, and what are the different compression formats supported?

Compression reduces the storage size of data and speeds up data transfer. Spark supports several compression formats:

- **Snappy**
- **Gzip**
- **Bzip2**
- **LZ4**
- **Zstandard (Zstd)**

62. Explain the filter transformation.

The `filter` transformation creates a new RDD by selecting only elements that satisfy a given predicate function.

63. How to start and stop Spark in the interactive shell?

To start Spark in the interactive shell:

- Use `spark-shell` for Scala or `pyspark` for Python. To stop Spark:
- Use `:quit` or `Ctrl + D` in the shell.

64. Explain the `sortByKey()` operation.

`sortByKey()` sorts an RDD of key-value pairs by the key in ascending or descending order.

65. Explain `distinct()`, `union()`, `intersection()`, and `subtract()` transformations in Spark.

- **distinct()**: Returns an RDD with duplicate elements removed.
- **union()**: Combines two RDDs into one.
- **intersection()**: Returns an RDD with elements common to both RDDs.
- **subtract()**: Returns an RDD with elements in one RDD but not in another.

66. Explain `foreach()` operation in Apache Spark.

`foreach()` applies a function to each element in the RDD, typically used for side effects like updating an external data store.

67. `groupByKey` vs `reduceByKey` in Apache Spark.

- **groupByKey**: Groups values by key and shuffles all data across the network, which can be less efficient.
- **reduceByKey**: Combines values for each key locally before shuffling, reducing network traffic.

68. Explain `mapPartitions()` and `mapPartitionsWithIndex()`.

- **mapPartitions()**: Applies a function to each partition of the RDD.
- **mapPartitionsWithIndex()**: Applies a function to each partition, providing the partition index.

69. What is `map` in Apache Spark?

`map` is a transformation that applies a function to each element in the RDD, resulting in a new RDD.

70. What is `flatMap` in Apache Spark?

`flatMap` is a transformation that applies a function to each element, resulting in multiple elements (a flat structure) for each input.

71. Explain `fold()` operation in Spark.

`fold()` aggregates the elements of an RDD using an associative function and a "zero value" (an initial value).

72. Explain `createOrReplaceTempView()` API.

`createOrReplaceTempView()` registers a DataFrame as a temporary table in Spark SQL, allowing it to be queried using SQL.

73. Explain `values()` operation in Apache Spark.

`values()` returns an RDD containing only the values of key-value pairs.

74. Explain `keys()` operation in Apache Spark.

`keys()` returns an RDD containing only the keys of key-value pairs.

75. Explain `textFile` vs `wholeTextFiles` in Spark.

- **textFile()**: Reads a text file and creates an RDD of strings, each representing a line.
- **wholeTextFiles()**: Reads entire files and creates an RDD of (filename, content) pairs.

76. Explain `cogroup()` operation in Spark.

`cogroup()` groups data from two or more RDDs sharing the same key.

77. Explain `pipe()` operation in Apache Spark.

`pipe()` passes each partition of an RDD to an external script or program and returns the output as an RDD.

78. Explain Spark `coalesce()` operation.

`coalesce()` reduces the number of partitions in an RDD, useful for minimizing shuffling when reducing the data size.

79. Explain the `repartition()` operation in Spark.

`repartition()` reshuffles data across partitions, increasing or decreasing the number of partitions, involving a full shuffle of data.

80. Explain `fullOuterJoin()` operation in Apache Spark.

`fullOuterJoin()` returns an RDD with all pairs of elements for matching keys and `null` for non-matching keys from both RDDs.

81. Explain Spark `leftOuterJoin()` and `rightOuterJoin()` operations.

- **leftOuterJoin()**: Returns all key-value pairs from the left RDD and matching pairs from the right, filling with `null` where no match is found.
- **rightOuterJoin()**: Returns all key-value pairs from the right RDD and matching pairs from the left, filling with `null` where no match is found.

82. Explain Spark `join()` operation.

`join()` returns an RDD with all pairs of elements with matching keys from both RDDs.

83. Explain `top()` and `takeOrdered()` operations.

- **top()**: Returns the top `n` elements from an RDD in descending order.
- **takeOrdered()**: Returns the top `n` elements from an RDD in ascending order.

84. Explain `first()` operation in Spark.

`first()` returns the first element of an RDD.

85. Explain `sum()`, `max()`, `min()` operations in Apache Spark.

These operations compute the sum, maximum, and minimum of elements in an RDD, respectively.

86. Explain `countByValue()` operation in Apache Spark RDD.

`countByValue()` returns a map of the counts of each unique value in the RDD.

87. Explain the `lookup()` operation in Spark.

`lookup()` returns the list of values associated with a given key in a paired RDD.

88. Explain Spark `countByKey()` operation.

`countByKey()` returns a map of the counts of each key in a paired RDD.

89. Explain Spark `saveAsTextFile()` operation.

`saveAsTextFile()` saves the RDD content as a text file or set of text files.

90. Explain `reduceByKey()` Spark operation.

`reduceByKey()` applies a reducing function to the elements with the same key, reducing them to a single element per key.

91. Explain the operation `reduce()` in Spark.

`reduce()` aggregates the elements of an RDD using an associative and commutative function.

92. Explain the action `count()` in Spark RDD.

`count()` returns the number of elements in an RDD.

93. Explain Spark `map()` transformation.

`map()` applies a function to each element of an RDD, creating a new RDD with the results.

94. Explain the `flatMap()` transformation in Apache Spark.

`flatMap()` applies a function that returns an iterable to each element and flattens the results into a single RDD.

95. What are the limitations of Apache Spark?

Limitations include high memory consumption, not ideal for OLTP (transactional processing), lack of a mature security framework, and dependency on cluster resources.

96. What is Spark SQL?

Spark SQL is a Spark module for structured data processing, providing a DataFrame API and allowing SQL queries to be executed.

97. Explain Spark SQL caching and uncaching.

- **Caching**: Storing DataFrames in memory for faster access.
- **Uncaching**: Removing cached DataFrames to free memory.

98. Explain Spark Streaming.

Spark Streaming is an extension of Spark for processing real-time data streams.

99. What is DStream in Apache Spark Streaming?

DStream (Discretized Stream) is a sequence of RDDs representing a continuous stream of data.

100. Explain different transformations in DStream in Apache Spark Streaming.

Transformations include:

- **map()**, **flatMap()**, **filter()**
- **reduceByKeyAndWindow()**
- **window()**, **countByWindow()**
- **updateStateByKey()**

101. What is the Starvation scenario in Spark Streaming?

Starvation occurs when all tasks are waiting for resources that are occupied by other long-running tasks, leading to delays or deadlocks.

102. Explain the level of parallelism in Spark Streaming.

Parallelism is controlled by the number of partitions in RDDs; increasing partitions increases the level of parallelism.

103. What are the different input sources for Spark Streaming?

Input sources include:

- Kafka
- Flume
- Kinesis
- Socket
- HDFS or S3

104. Explain Spark Streaming with Socket.

Spark Streaming can receive real-time data streams over a socket using `socketTextStream()`.

105. Define the roles of the file system in any framework.

The file system manages data storage, access, and security, ensuring data integrity and availability.

106. How do you parse data in XML? Which kind of class do you use with Java to parse data?

To parse XML data in Java, you can use classes from the `javax.xml.parsers` package, such as:

- **DocumentBuilder**: Used with the Document Object Model (DOM) for in-memory tree representation.
- **SAXParser**: Used with the Simple API for XML (SAX) for event-driven parsing.

107. What is PageRank in Spark?

PageRank is an algorithm used to rank web pages in search engine results, based on the number and quality of links to a page. In Spark, it can be implemented using RDDs or DataFrames to compute the rank of nodes in a graph.

108. What are the roles and responsibilities of worker nodes in the Apache Spark cluster? Is the Worker Node in Spark the same as the Slave Node?

- **Worker Nodes**: Execute tasks assigned by the Spark Driver, manage executors, and store data in memory or disk as required.
- **Slave Nodes**: Worker nodes in Spark are commonly referred to as slave nodes. Both terms are used interchangeably.

109. How to split a single HDFS block into partitions in an RDD?

When reading from HDFS, Spark splits a single block into multiple partitions based on the number of available cores or executors. You can also use the `repartition()` method to explicitly specify the number of partitions.

110. On what basis can you differentiate RDD, DataFrame, and DataSet?

- **RDD**: Low-level, unstructured data; provides functional programming APIs.
- **DataFrame**: Higher-level abstraction with schema; optimized for SQL queries and transformations.
- **Dataset**: Combines features of RDDs and DataFrames; offers type safety and object-oriented programming.