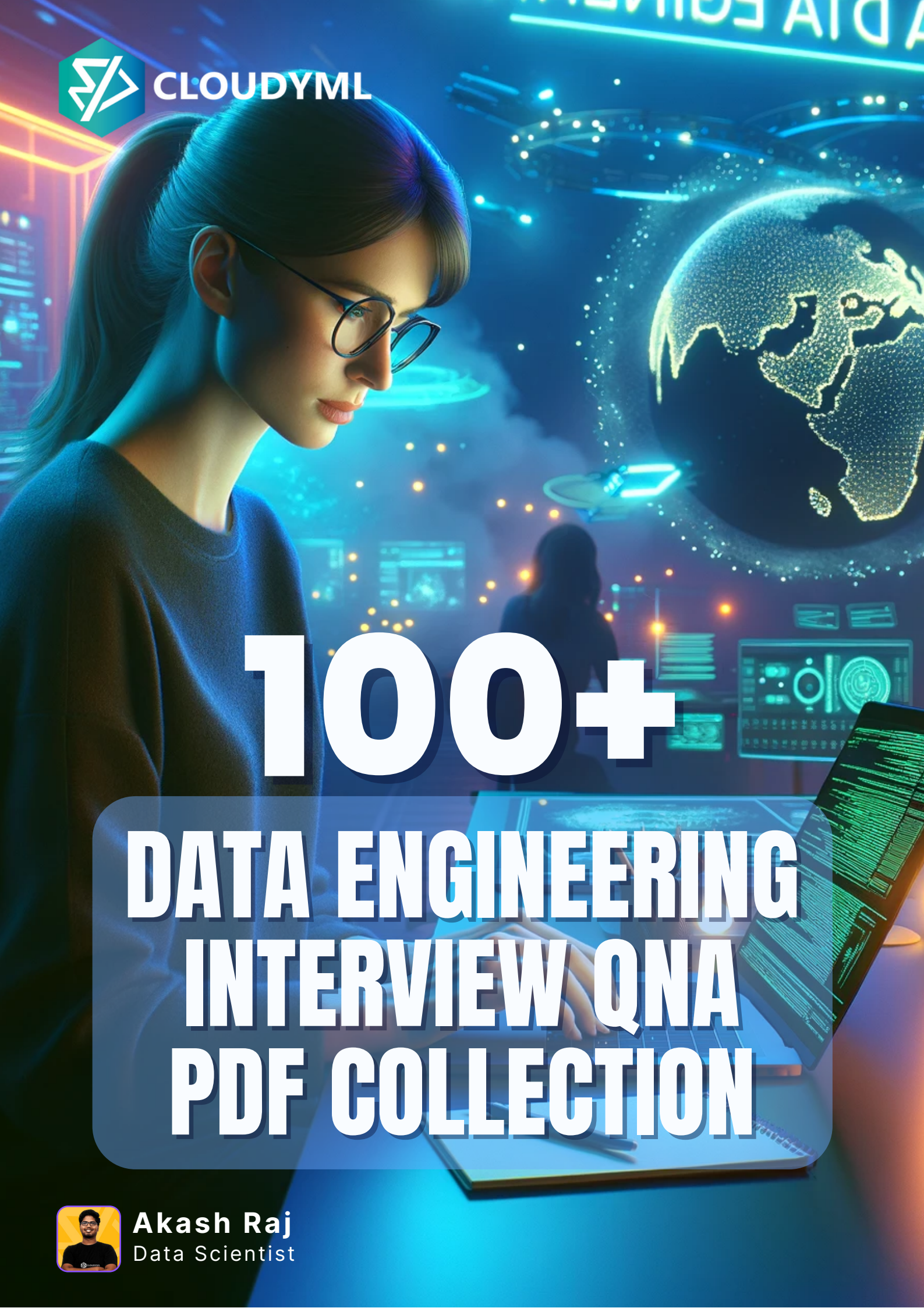

Akash Raj
Data Scientist

1.What applications are supported by Hive?

Hive is primarily used for querying and analyzing large datasets stored in Hadoop Distributed File System (HDFS). It is often used for data warehousing, ETL (Extract, Transform, Load) processes, and data analysis. Common use cases include log processing, data aggregation, and data exploration.

2.What is Fault Tolerance in a Data Node?

Data nodes are made fault-tolerant through data replication. HDFS replicates data blocks across multiple data nodes to ensure data availability in case of node failures.

3.What is Fault Tolerance in Name Nodes?

The NameNode in HDFS achieves fault tolerance through the Secondary NameNode, which periodically creates checkpoints of the namespace. In Hadoop 2.x and later, HDFS also supports HA (High Availability) with multiple NameNodes.


Akash Raj
@cloudyml.akash

4.What are the different types of tables available in Hive?

Hive supports two main types of tables: Managed Tables and External Tables. Managed tables store data in a Hive-specific directory, while external tables reference data stored outside of Hive's control, such as in HDFS or other storage systems.

5.What is the difference between managed and external tables?

Managed tables have their data managed and controlled by Hive. Hive assumes full responsibility for the data's lifecycle, including deleting data when the table is dropped. External tables, on the other hand, don't manage the data themselves and only maintain metadata references to the external data location.

6.How to Update a Record in HBase Table?

To update a record in an HBase table, you typically perform a new write operation with the updated data for the same row key. HBase will overwrite the existing data with the new data.


Akash Raj
@cloudyml.akash

10.How to achieve High Availability of Name Node?

High availability of the NameNode is achieved through HA configurations, where multiple NameNodes run in an active-standby setup. ZooKeeper is often used to manage the failover.

11.Determine the Significance of Counters in MapReduce.

Counters in MapReduce are used to keep track of various statistics during job execution, such as the number of records processed, custom metrics, or errors encountered.

12.Where does the data of a Hive table get stored?

In the case of managed tables, the data is stored in a directory managed by Hive within HDFS. For external tables, the data remains in its original location, and Hive maintains metadata references to it.

7.What are Issues with Lots of Small Files in HDFS?

Storing lots of small files in HDFS can lead to inefficiencies in terms of storage and metadata management. It can impact overall cluster performance.

8.How to Overcome Issues with Lots of Small Files?

To overcome issues with many small files, you can use techniques like Hadoop Archives (HAR), SequenceFiles, or CombineFileInputFormat to bundle small files together.

9.What are the types of Metastore in Hive?

The Block Scanner in HDFS periodically scans data blocks on data nodes to detect and correct corruption. It helps ensure data integrity.

13.What is the purpose of Setting Number of Reducer Tasks to Zero?

Setting the number of reducer tasks to zero effectively disables the reduce phase of a MapReduce job. This is used when you want only the map phase and no data aggregation.

14.Location of Data in Hive Table?

The data of a Hive table is stored in HDFS, typically in the /user/hive/warehouse directory.

15.Why is HDFS Not Used by Hive Metastore for Storage?

Hive Metastore typically doesn't use HDFS for storage because the metastore itself stores metadata about tables, columns, partitions, and schemas, rather than actual data. Storing metadata in a distributed file system like HDFS is unnecessary and less efficient.

16.Difference Between External and Managed Tables in Hive:

- External Table: Data is stored externally, and Hive only manages the metadata. Dropping the table does not delete the data.
- Managed Table: Hive manages both the metadata and the data. Dropping the table also deletes the associated data.

17.Can Hive be used in OLTP systems?

Hive is not suitable for OLTP (Online Transaction Processing) systems because it is optimized for batch processing and is not designed for low-latency, high-concurrency transactional operations. OLAP (Online Analytical Processing) workloads are more suitable for Hive.

18.Can a table name be changed in Hive?

Yes, you can rename a table in Hive using the ALTER TABLE statement.

19.Can the default location of a managed table be changed in Hive?

Yes, you can specify a custom location for managed tables during table creation or alter the table to change its location using the LOCATION keyword in the CREATE TABLE or ALTER TABLE statements.

20.What is a Hive Metastore?

The Hive Metastore is a centralized metadata repository in Hive that stores metadata information about tables, partitions, columns, and other objects in the Hive data warehouse. It helps Hive manage and organize data effectively.

21.What are the types of Metastore in Hive?

There are two types of Hive Metastores: Local Metastore, which stores metadata on the local file system, and Remote Metastore, which stores metadata in a separate database like MySQL, PostgreSQL, or Oracle.

CLOUDYML

GET PLACED AS A

DATA ANALYST
DATA SCIENTIST
DATA ENGINEER

Become Master Of All The Data Tools

VISIT - WWW.CLOUDYML.COM

INDIA'S #1 MOST AFFORDABLE INDUSTRY ORIENTED COURSE PROVIDERS

22.What is the difference between Local and Remote Metastore?

The key difference is where metadata is stored. Local Metastore stores metadata on the local file system, which is suitable for small-scale setups. Remote Metastore stores metadata in a separate database, making it more scalable and suitable for larger deployments with multiple Hive instances.

23.What are the three different modes in which Hive can be operated?

Hive can operate in three different modes: Local Mode, MapReduce Mode, and Tez Mode. Local Mode is used for development and debugging, MapReduce Mode leverages Hadoop MapReduce for query execution, and Tez Mode uses the Apache Tez execution framework for improved performance.

24.Why is partitioning used in Hive?

Partitioning is used in Hive to improve query performance and manage data efficiently. It divides large tables into smaller, more manageable parts based on one or more columns, making it easier to filter and access specific data subsets.

25.What is dynamic partitioning and when is it used in Hive?

Dynamic partitioning in Hive is a feature that automatically creates partitions based on the values in specified columns during data insertion. It is useful when you want to create partitions on-the-fly without explicitly defining them in advance.

26.What are the Hive collection data types?

Hive supports complex data types, including ARRAY, MAP, and STRUCT for representing collections of data. These data types enable users to work with more structured and nested data in Hive tables.

27.What is Bucketing in Hive?

Bucketing is a technique in Hive where data is distributed into a fixed number of buckets based on a specific column's values. It is used to improve data organization and optimize certain types of queries, especially those involving joins.

28.How is bucketing helpful?

Bucketing helps optimize Hive queries by reducing data shuffling during join operations. It allows for more efficient data retrieval and can significantly improve query performance, especially when dealing with large datasets and complex joins.

29.When to Use Sort By and Order By in Hive

- Use SORT BY when you want to sort data within each reducer before writing it to the final output. It's more efficient for large datasets.
- Use ORDER BY when you need global sorting across all data, but be aware that it can be more resource-intensive.

30.Difference Between Partition and Bucket in Hive.

- Partition: Data is divided into directories based on the values of one or more columns. It's used for data organization and efficient filtering.
- Bucket: Data is divided into fixed-size buckets based on a hash function applied to one or more columns. It's used for optimization and faster querying.

31.Explain Dynamic Partitioning in Hive.

Dynamic partitioning in Hive is a feature that allows Hive to automatically create partitions during data insertion. It's used when you want to create partitions based on specific column values without predefining them.

32.How Hive Distributes Rows into Buckets?

Hive distributes rows into buckets using a hash function applied to one or more columns specified during table creation. The hash function determines which bucket each row goes into based on its values.

33.How to find a particular text or name in your HDFS?

You can find a particular text or name in the Hadoop Distributed File System (HDFS) by using the grep command. For example, you can run hadoop fs -cat /path/to/file | grep "search_text" to search for "search_text" in a specific file in HDFS.

34.Explain the Ingestion process.

Ingestion is the process of collecting, importing, and transferring data from various sources into a data storage system or data lake. It involves data extraction, transformation, and loading (ETL) operations to ensure that the data is in a format suitable for analysis. Ingestion processes can vary depending on the data sources and the target data storage technology being used.

35.What are HBase basics?

HBase is a distributed, column-oriented NoSQL database that runs on top of the Hadoop Distributed File System (HDFS). It is designed for storing and managing large amounts of sparse data. HBase is known for its scalability, real-time read and write capabilities, and is often used for applications that require low-latency access to data, such as time-series data and sensor data.

36.Explain SortMerge Bucket join.

SortMerge Bucket join is a type of join optimization technique in Apache Hive. It optimizes join operations by sorting and partitioning the data being joined based on a common key (bucketed column) and then performing the join operation. This helps reduce data shuffling and improves query performance.

37.What are Cache and Persist in Spark?

- cache() and persist() are methods in Spark that allow you to store RDD or DataFrame data in memory for faster access.
- cache() stores data in memory, and persist() allows you to specify different storage levels (e.g., MEMORY_ONLY, MEMORY_AND_DISK).

38.Difference between Repartition and Coalesce.

- repartition() is used to increase or decrease the number of partitions in an RDD or DataFrame. It involves shuffling.
- coalesce() reduces the number of partitions without shuffling, making it more efficient for reducing the number of partitions.

39.Difference between SparkContext and SparkSession.

- SparkContext is the entry point for low-level Spark functionality, such as creating RDDs and managing cluster resources.
- SparkSession is a higher-level interface that provides a unified entry point for working with structured data, including DataFrames and Datasets, and is commonly used for SQL queries.

40.Difference between Spark and MapReduce.

Processing Model: Spark uses an in-memory processing model, while MapReduce relies on disk storage for intermediate data.
Performance: Spark is generally faster than MapReduce due to its ability to cache data in memory.
Ease of Use: Spark provides high-level APIs in various languages (e.g., Scala, Python), making it more user-friendly.
Data Processing: Spark supports batch, real-time, and iterative processing, whereas MapReduce primarily focuses on batch processing.

41.What is ROD (Resilient Distributed Dataset)?

ROD is not a standard term. It might be a typo or an abbreviation for RDD (Resilient Distributed Dataset), which is a fundamental data structure in Spark.

42.What are Key Features of Apache Spark?

Key features of Apache Spark include in-memory data processing, support for various data sources, easy-to-use APIs (including DataFrames and Datasets), fault tolerance, and support for batch, streaming, and machine learning workloads.

43.What are Components of the Spark Ecosystem?

The Spark ecosystem includes components like Spark Core, Spark SQL, Spark Streaming, MLlib (machine learning library), GraphX, and SparkR.

44.What is RDD Lineage?

RDD lineage is a directed acyclic graph (DAG) that represents the sequence of transformations applied to an RDD. It enables fault tolerance and recovery in Spark by allowing the recreation of lost data through transformations.

45.WHat is DStream?

DStream (Discretized Stream) is a high-level abstraction in Spark Streaming for processing real-time data streams. It represents a sequence of RDDs.

46.Explain the architecture of Spark.

Spark has a master/worker architecture:
Driver Program: Coordinates the application and schedules tasks.
Cluster Manager: Manages resources across the cluster (e.g., YARN, Mesos).
Executors: Execute tasks and store data in-memory or on disk.
Cluster: A group of machines where Spark runs.

47.How Spark achieves fault tolerance?

Spark achieves fault tolerance through lineage information. It tracks the transformations applied to the original data to recreate lost partitions in case of node failures.

48.What is RDD (Resilient Distributed Dataset)?

RDD is a fundamental data structure in Spark. It's an immutable, distributed collection of data that can be processed in parallel. RDDs offer fault tolerance through lineage and can be cached in memory for faster processing.

49. What is transformation and action in Spark?

Transformation: Transformations are operations that create a new RDD from an existing one, like map, filter, and reduceByKey. They are lazily evaluated.
Action: Actions are operations that trigger the execution of transformations and return a result to the driver program, like count, collect, and saveAsTextFile.

50.Difference between DAG (Directed Acyclic Graph) and Lineage.

DAG: DAG represents the logical execution plan of Spark operations, showing their dependencies.
Lineage: Lineage is the physical execution plan that describes how to recover lost RDD partitions in case of failures. It's derived from the DAG.

51.What is partitioning in Spark and how does Spark partition data?

Partitioning is the process of dividing data into smaller, manageable chunks. In Spark, data is partitioned in RDDs, and the number of partitions determines the parallelism of the job. You can control partitioning explicitly when loading data or through transformations like repartition and coalesce.

52.Difference between narrow and wide transformations.

Narrow Transformation: Narrow transformations do not require data shuffling across partitions. Examples include map and filter.
Wide Transformation: Wide transformations involve data shuffling between partitions, like groupBy and reduceByKey. They trigger a stage boundary in Spark's execution plan.

53.Briefly explain the different cluster managers available in Apache Spark.

Apache Spark can be integrated with various cluster managers like Apache Hadoop YARN, Apache Mesos, and Standalone Spark Cluster Manager. These managers allocate and manage cluster resources for Spark applications.

54.Explain coalescing in Spark with an example.

Coalescing is a transformation in Spark that reduces the number of partitions while minimizing data movement. For example, if you have 100 partitions but only need 10, you can use coalesce(10) to coalesce them into 10 partitions.

55.How to calculate executor memory in Spark?

Executor memory is typically calculated by considering the total memory available on each worker node and dividing it by the number of executors. It's important to leave some memory for the OS and other processes. The formula might look like: (Total Memory - Reserved Memory) / Number of Executors.

56.Importance of sliding window operation in Spark.

Sliding window operations are crucial in stream processing applications. They allow you to analyze data over a rolling time or event window, making it possible to calculate trends, aggregations, and patterns in real-time data.

57.Different levels of persistence in Spark.

Spark offers various persistence levels for caching RDDs or DataFrames, including MEMORY_ONLY, MEMORY_ONLY_SER, MEMORY_AND_DISK, and more. Each level balances between memory usage and performance.

58.How to calculate executor memory in Spark?

Executor memory is typically calculated by considering the total memory available on each worker node and dividing it by the number of executors. It's important to leave some memory for the OS and other processes. The formula might look like: (Total Memory - Reserved Memory) / Number of Executors.

59.Importance of sliding window operation in Spark.

Sliding window operations are crucial in stream processing applications. They allow you to analyze data over a rolling time or event window, making it possible to calculate trends, aggregations, and patterns in real-time data.

60.Different levels of persistence in Spark.

Spark offers various persistence levels for caching RDDs or DataFrames, including MEMORY_ONLY, MEMORY_ONLY_SER, MEMORY_AND_DISK, and more. Each level balances between memory usage and performance.

61.How to join two large tables in Spark.

When joining large tables in Spark, consider using broadcast joins for smaller tables and shuffle joins for larger tables. Broadcasting smaller tables can reduce data shuffling and improve performance.

62.How to read Parquet file format in Spark?

You can read Parquet files in Spark using the spark.read.parquet("path/to/parquet") method. Spark provides built-in support for Parquet files, making it easy to read and process data stored in this columnar storage format.

63.Difference between SQL and NoSQL.

SQL (Structured Query Language) and NoSQL (Not Only SQL) are two different types of database management systems. SQL databases are relational databases that use structured tables and schemas for data storage, while NoSQL databases are non-relational and can handle unstructured or semi-structured data. SQL databases are known for their ACID compliance, while NoSQL databases are typically used for scalability, flexibility, and handling large volumes of data.

64.Who will run the Spark job in your team?

In our team, the responsibility for running Spark jobs is typically shared among team members based on a task's ownership or expertise. This can include data engineers, data scientists, or DevOps engineers, depending on the nature of the Spark job and the specific requirements.

65.Explain Tungsten.

Tungsten is an optimization framework in Apache Spark designed to improve the performance and efficiency of Spark applications. It includes several enhancements like memory management improvements, code generation, and query optimization. Tungsten aims to make Spark run faster and use resources more efficiently.

66.How to join two bigger tables in Spark.

To join two large tables in Spark, you can use the join operation provided by Spark's DataFrame API. It's important to choose the appropriate join type (e.g., inner join, outer join) and optimize the join by using techniques like broadcasting smaller tables or partitioning and bucketing data for efficient processing.

CLOUDYML

DATA SUPERSTAR

The Highest Paying Job Role Of 21st Century

- Get Practical Learning Experience
- 50+ Industry Oriented Projects
- 1-1 Doubt Clearance Support
- Industrial Internship Opportunity
- 100% Placement Guarantee

Learn From Scratch
@ Most Affordable Price

VISIT OUR WEBSITE

WWW.CLOUDYML.COM

Trusted By 18,000+ Students From India, USA, UK, Canada, Germany, Australia & 20+ Other Countries

67.Explain Left Outer Join.

A Left Outer Join, also known as a Left Join, is a type of database join operation. It returns all the rows from the left table (table1) and the matching rows from the right table (table2). If there are no matches in the right table, NULL values are returned for columns from the right table.

68.How to count the lines in a file using a Linux command.

You can count the lines in a file using the wc (word count) command in Linux. For example, to count lines in a file named "example.txt," you can run wc -l example.txt.

69.How to achieve map-side joins in Hive.

Map-side joins in Hive involve using the MAPJOIN hint to instruct Hive to perform a join operation at the map phase, reducing the need for data shuffling. This is suitable when one of the tables is small enough to fit in memory.

70.Will it go to reducer if using a select command in Hive?

No, when you execute a simple SELECT command in Hive, it does not involve a MapReduce job or reducers. Select queries typically retrieve data directly from the HDFS or Hive tables without any MapReduce processing.

71.How to validate data once ingestion is done?

Data validation after ingestion involves checking data quality, completeness, and accuracy. It can be done using scripts or tools to perform checks like schema validation, data profiling, duplicate detection, and outlier detection.

72.Use the split-by command in Sqoop?

The --split-by option in Sqoop is used to specify a column that Sqoop will use to split the data into multiple parallel threads during import. This helps improve the import performance, especially when dealing with large datasets.

73.Difference between DataFrame and Datasets.

DataFrames and Datasets are both abstractions in Apache Spark for working with structured data. DataFrames are a distributed collection of data organized into named columns and rows, while Datasets are a strongly typed, distributed collection of data that combines the features of DataFrames and RDDs. Datasets provide type-safety and are available in Scala and Java.

74.Schema on Read vs. Schema on Write.

Schema on Read and Schema on Write are approaches to handling data in a data storage system. Schema on Write involves defining and enforcing a schema (structure) for data at the time of ingestion into the storage system. Schema on Read, on the other hand, allows data to be ingested without a predefined schema and applies the schema when the data is read or queried. Hadoop and many NoSQL databases often use Schema on Read, providing flexibility but requiring schema enforcement at query time. Relational databases typically use Schema on Write, ensuring data consistency but requiring schema changes before ingestion.

75.Different Types of Partitioning in Hive?

Hive supports several types of partitioning, including:
Static Partitioning: Where partitions are explicitly defined and assigned during data loading.
Dynamic Partitioning: Automatically creates partitions based on column values during data insertion.
Bucketing: Divides data into fixed-size buckets based on a hashing algorithm, improving query performance.
Virtual Partitioning: Allows logical partitions without physically rearranging data on disk.

76. SQL Query to Find Counts Based on Age Group?

To find counts based on age groups in SQL, you can use a query like:
sql
Copy code

```
SELECT age_group, COUNT(*) as count
FROM (
  SELECT CASE
        WHEN age BETWEEN 0 AND 18 THEN '0-18'
        WHEN age BETWEEN 19 AND 30 THEN '19-30'
        WHEN age BETWEEN 31 AND 45 THEN '31-45'
        ELSE '46+'
      END AS age_group
  FROM your_table
) AS age_groups
GROUP BY age_group;
```

77. What is an Executor Node in Spark?

An Executor Node is a worker node in an Apache Spark cluster responsible for running tasks for a Spark application. Each executor runs on a separate JVM and manages data storage and computation. Executors execute tasks in parallel across the cluster and communicate with the driver program.

78.Difference Between Hadoop & Spark.

Hadoop and Spark are both big data processing frameworks, but they have differences. Hadoop is primarily designed for batch processing, while Spark supports batch, real-time, and iterative processing. Spark also maintains data in-memory, offering faster processing than Hadoop's disk-based storage.

79.How to Find Duplicate Values in SQL?

You can find duplicate values in SQL using a query with the GROUP BY clause and the HAVING clause. For example:
sql
Copy code

```
SELECT column_name, COUNT(*)
FROM your_table
GROUP BY column_name
HAVING COUNT(*) > 1;
```

80.Difference Between Row Number and Dense_rank Method in SQL.

ROW_NUMBER() assigns a unique integer to each row, starting from 1 for each partition.
DENSE_RANK() assigns a rank to rows within a partition, and the same rank can be assigned to multiple rows if they have the same values

81.How to Find the 2nd Largest Number in the Table?

You can find the 2nd largest number in SQL using a query like:
sql
Copy code
```
SELECT DISTINCT column_name
FROM your_table
ORDER BY column_name DESC
LIMIT 1 OFFSET 1;
```

82. State CAP Theorem.

CAP Theorem is a concept in distributed systems. It states that in a distributed system, you can have at most two out of three guarantees: Consistency, Availability, and Partition Tolerance. This means that during a network partition (P), you must choose between maintaining Consistency (C) or Availability (A).

83. How to Achieve Map Side Joins in Hive?

To achieve Map Side Joins in Hive, you can use the MAPJOIN hint in your SQL query. This instructs Hive to perform a join at the map phase, which is efficient when one of the tables is small enough to fit in memory.

84. Give File Formats Used in Your Project.

The choice of file formats depends on the project's requirements, but common formats include Parquet, ORC, Avro, CSV, and JSON. Parquet and ORC are columnar storage formats known for their compression and query performance benefits.

85.Difference Between List and Tuple.

In Python, lists are mutable, ordered collections of elements, while tuples are immutable, ordered collections. Lists are defined using square brackets [ ], and tuples use parentheses ( ).

86. What are Various Hive Optimization Techniques:

Hive optimization techniques include:

Partitioning and Bucketing.
Using ORC or Parquet file formats.
Using Vectorized Query Execution.
Cost-based optimization with Tez or Spark.
Caching intermediate results.
Using indexes.

87. Give File Formats Used in Your Project.

The choice of file formats depends on the project's requirements, but common formats include Parquet, ORC, Avro, CSV, and JSON. Parquet and ORC are columnar storage formats known for their compression and query performance benefits.

88.How MapReduce Works?

MapReduce is a programming model for processing and generating large datasets. It divides a job into two phases: the Map phase (data processing) and the Reduce phase (aggregation). Mappers process data in parallel, and the results are shuffled and grouped by keys before reducers aggregate the results.

89. What are types of Tables in Hive:

Hive supports various table types, including Managed Tables (Hive manages data and metadata), External Tables (data managed outside Hive), and Temporary Tables (session-specific).

90.How to Drop a Table in HBase?

You can drop a table in HBase using the HBase shell by running the disable 'table_name' command followed by drop 'table_name'.

91. What is SerDe in Hive?

SerDe stands for Serializer/Deserializer. In Hive, SerDe is a Java library used to serialize data before writing it to HDFS and deserialize data when reading it from HDFS. It defines the structure of data within Hive tables, allowing Hive to work with various data formats like JSON, Avro, and XML.

93.Why Dataset is Preferred Compared to DataFrame?

Datasets are preferred over DataFrames when you need strong typing and the benefits of both RDDs and DataFrames. Datasets allow for type-safe, functional-style operations while providing optimizations like Catalyst and Tungsten. DataFrames, on the other hand, have less type safety.

94.How to Check File Size in Hadoop?

You can check the file size in Hadoop using the hadoop fs -du command. For example, hadoop fs -du -h /path/to/file_or_directory will display the file or directory size in human-readable format.

95.How to Submit a Spark Job?

You can submit a Spark job using the spark-submit command, specifying your application's main class and any required configuration options. For example:

96.Why Dataset is Preferred Compared to DataFrame?

Datasets are preferred over DataFrames when you need strong typing and the benefits of both RDDs and DataFrames. Datasets allow for type-safe, functional-style operations while providing optimizations like Catalyst and Tungsten. DataFrames, on the other hand, have less type safety.

97.How to Check File Size in Hadoop?

You can check the file size in Hadoop using the hadoop fs -du command. For example, hadoop fs -du -h /path/to/file_or_directory will display the file or directory size in human-readable format.

98.How to Submit a Spark Job?

You can submit a Spark job using the spark-submit command, specifying your application's main class and any required configuration options. For example:

spark-submit --class your_main_class --master yarn --deploy-mode cluster your_app.jar

99. What is Vectorization and Why It's Used?

Vectorization is a technique used to perform batch operations on a set of data instead of processing one element at a time. It's used in Spark and other systems to improve performance by minimizing the overhead of looping through individual records.

100. What is Sampling in Hive?

Sampling in Hive involves selecting a subset of data from a large dataset for analysis or testing purposes. Hive provides the TABLESAMPLE clause that allows you to sample data based on a percentage or number of rows.

101.Different Types of XMI Files in Hadoop.

XMI (XML Metadata Interchange) files are not typically associated with Hadoop. Hadoop primarily deals with data storage and processing. XMI files are used for exchanging metadata between software modeling tools.

SCALA INTERVIEW QUESTIONS

102.What is a Case Class?

In Scala, a case class is a class that is used primarily for storing data. It automatically provides various useful methods like equals, hashCode, and a default toString. Case classes are often used for modeling immutable data structures.

103.What is a Scala Trait?

A Scala trait is similar to an interface in other programming languages. It defines a set of abstract methods that can be mixed into classes to provide additional behavior. Traits can also contain concrete methods and fields.

104.What is a Higher-Order Function and Example?

A higher-order function is a function that takes one or more functions as arguments or returns a function as its result. An example in Scala:
scala def applyFunction(func: Int => Int, x: Int): Int = func(x) val square = (x: Int) => x * x val result = applyFunction(square, 5) // result will be 25

105.What is a Companion Object?

In Scala, a companion object is an object with the same name as a class or trait. It often contains static methods and properties related to the class or trait. Companion objects are used for encapsulating utilities and common behaviors.

106.Difference Between DAG and Lineage Graph.

- DAG (Directed Acyclic Graph) represents the logical flow of a Spark application's stages and tasks.
- Lineage Graph, in the context of Spark, represents the lineage of a Resilient Distributed Dataset (RDD), showing how it is derived from other RDDs through transformations.

107.What is Catalyst Optimizer?

Catalyst is the query optimizer framework in Apache Spark. It optimizes query plans for better performance by applying various rule-based and cost-based optimizations.

108. What are techniques to tune Your Spark Application?

Techniques include optimizing Spark configurations, using appropriate data formats and storage, tuning the number of partitions, caching, and leveraging broadcast joins, among others.

109.Difference Between Broadcast and Accumulators.

- Broadcast is used to share read-only variables with worker nodes efficiently.
- Accumulators are used for aggregating values from worker nodes back to the driver in a parallel and fault-tolerant way.

110.What is Broadcast Join?

Catalyst is the query optimizer framework in Apache Spark. It optimizes query plans for better performance by applying various rule-based and cost-based optimizations.

111.Difference Between SparkContext and SparkSession.

- SparkContext is the entry point to Spark and represents the connection to the Spark cluster.
- SparkSession is higher-level and includes a SparkContext. It provides a unified interface for working with data in Spark and is commonly used for SQL, DataFrames, and Datasets.

112. What are Types of Transformations in Spark and Differences?

Transformations in Spark include map, filter, reduce, groupByKey, flatMap, and more. Transformations are generally divided into narrow (produce one-to-one mapping) and wide (shuffle) transformations based on their impact on data partitioning.

113. What are Operations of Data Frame?

DataFrames support various operations like select, filter, groupBy, agg, join, orderBy, distinct, and more, which allow you to manipulate and analyze data efficiently.

114.Why People are Going with Spark and Not MapReduce?

- Spark is faster: It performs in-memory processing, reducing the need to read and write to disk, making it faster than MapReduce.
- Ease of use: Spark provides high-level APIs in various languages, making it more accessible to developers.
- Versatility: Spark supports batch, real-time, and machine learning workloads in a single framework.
- Fault tolerance: Spark has built-in fault tolerance mechanisms.

115.Difference Between Partitioning and Bucketing.

- Partitioning: It divides data into directories or subdirectories based on one or more columns' values. Each partition represents a subset of the data, making it easy to filter data by partition keys.
- Bucketing: It divides data into fixed-size buckets based on a hashing algorithm applied to one or more columns. Bucketing is used for optimization, especially in Hive, to improve query performance.

116. What are ways of Handling Incremental Data?

Handling incremental data involves identifying new or changed records since the last processing run. Techniques include using timestamps or flags in the data, tracking record versions, or using watermarking in streaming data.

117. Explain YARN (Yet Another Resource Negotiator) Architecture.

YARN is the resource management layer in Hadoop. It consists of a ResourceManager (RM) for managing cluster resources and ApplicationMasters (AM) for managing application-specific tasks. YARN allows multiple applications to run on a Hadoop cluster simultaneously.

118. What is an Incremental Sqoop?

Incremental Sqoop is a feature that allows you to import only new or changed records from a database into Hadoop. You can achieve this by specifying the --incremental flag along with the --check-column option, indicating the column to check for changes.

119.Why Use HBase and How It Stores Data?

HBase is used for its real-time, random read/write capabilities, making it suitable for applications like sensor data, social media, and recommendation engines. It stores data in tables with rows identified by unique RowKeys. Data is stored in sorted order, and it uses HDFS for distributed storage.

120.What are various optimization Techniques in Hive?

Hive optimization techniques include using appropriate file formats (ORC, Parquet), bucketing, partitioning, indexing, cost-based optimization with Tez or Spark, vectorization, and enabling predicate pushdown.