**1. Features of WebHDFS**

WebHDFS is a RESTful API that provides HTTP access to HDFS (Hadoop Distributed File System). Key features include:

1. **REST API**: Allows applications to interact with HDFS over HTTP.
2. **Authentication**: Supports Kerberos, Pseudo, and SSL for secure communication.
3. **File Operations**: Enables reading, writing, deleting, and appending to files in HDFS.
4. **Directory Operations**: Supports creating, listing, and deleting directories.
5. **Data Transfer**: Facilitates data transfer between the client and HDFS, allowing both single and multipart uploads.

**2. Use of CURL Command**

The curl command is used to transfer data to or from a server using various protocols, including HTTP. In the context of WebHDFS, curl can interact with HDFS over HTTP.

**Example**: Reading a file from HDFS using WebHDFS and curl.

curl -i -L -X GET "http://<namenode\_host>:<port>/webhdfs/v1/<path\_to\_file>?op=OPEN"

This command sends an HTTP GET request to the WebHDFS API to open and read the file located at <path\_to\_file>.

**3. Use of Sqoop**

Sqoop (SQL-to-Hadoop) is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases.

**Example**: Importing data from a MySQL database to HDFS.

sqoop import --connect jdbc:mysql://hostname:port/database\_name \

--username username \

--password password \

--table table\_name \

--target-dir /path/to/hdfs/dir

This command imports data from the specified MySQL table into HDFS.

**4. Three Main Phases of a MapReduce Job**

1. **Map Phase**: Processes input data and generates intermediate key-value pairs.
2. **Shuffle and Sort Phase**: Transfers and sorts the intermediate key-value pairs.
3. **Reduce Phase**: Processes sorted key-value pairs to produce the final output.

**5. Four Main Components of YARN**

1. **ResourceManager**: Manages cluster resources and schedules applications.
2. **NodeManager**: Manages resources on individual nodes and monitors container resource usage.
3. **ApplicationMaster**: Manages the lifecycle of an application and coordinates with ResourceManager and NodeManagers.
4. **Container**: An isolated unit of computation, including resources like CPU, memory, and disk, where tasks are executed.

**6. What Happens if All <key, value> Pairs Output by a Mapper Do Not Fit into the Memory of the Mapper?**

If all the <key, value> pairs output by a Mapper do not fit into memory, Hadoop writes the data to local disk in a buffer and spills it to disk when the buffer reaches a threshold. These spills are merged and sorted to form the final output of the Mapper.

**7. Use of Combiner in MapReduce**

The Combiner function acts as a mini-reducer, processing the Mapper output to reduce the volume of data before transferring it to the Reducer. This optimization step helps minimize data transfer across the network.

**Example**: Summing intermediate values in a word count job.

**8. Use of Partitioner in MapReduce**

The Partitioner determines which reducer will process a particular key. This ensures an even distribution of data across reducers, which helps achieve load balancing.

**Example**: HashPartitioner is the default partitioner which uses the hash code of the key to distribute keys among reducers.

**9. What is Hive?**

Hive is a data warehouse infrastructure built on top of Hadoop. It provides data summarization, query, and analysis capabilities through a SQL-like language called HiveQL.

**10. Partitioning in Hive**

Partitioning in Hive divides a table into parts based on the values of a specific column, improving query performance.

**Example**: Partitioning a sales table by year and month.

CREATE TABLE sales (

item\_id INT,

amount DECIMAL(10, 2)

)

PARTITIONED BY (year INT, month INT);

**11. Bucketing in Hive**

Bucketing groups data into fixed-size buckets based on the hash function of a column, optimizing query performance on that column.

**Example**: Bucketing a table by user ID into 10 buckets.

CREATE TABLE user\_logs (

user\_id INT,

activity STRING

)

CLUSTERED BY (user\_id) INTO 10 BUCKETS;

**12. Data Sorting in Hive**

Data sorting in Hive is achieved using the ORDER BY, SORT BY, DISTRIBUTE BY, and CLUSTER BY clauses.

* **ORDER BY**: Sorts the entire result set.
* **SORT BY**: Sorts data within each reducer.
* **DISTRIBUTE BY**: Divides data based on column values.
* **CLUSTER BY**: Distributes and sorts data.

**13. Join Strategies in Hive**

1. **Map Join**: Small tables are loaded into memory and joined with a larger table.
2. **Sort-Merge Join**: Input tables are sorted and merged.
3. **Bucket Map Join**: Tables are bucketed and the join is performed in a map phase.

**Example**: Performing a join operation in Hive.

SELECT a.\*, b.\*

FROM table\_a a

JOIN table\_b b

ON a.id = b.id;

Each join strategy optimizes the performance based on table size and structure.