Spark

1. Spark - In out of memory issue, how will you decide to which parameter to increase or decrease

- The out of memory issue can occur at driver level and executor level

  we can solve this problem proper limit using spark.driver.maxResultSize

  by default spark.driver.maxResultSize is 1gb

Broadcast Join - when performing the broadcast joins the table is first materialized at the driver side and then broadcasted to the executors. If there are multiple table are being broadcasted we may get out of memory issue

to resolve this increase the memory of driver or reduce the value of spark.sql.autoBroadcastJoinThreshold

2. Spark – What is the purpose of repartition?

repartition - This method is use to increase or decrease the number of partitions

more shuffle will happen while using the repartition as compare to coalesce

It will create approximately equal size of partitions

3. Spark – What is the difference between Repartition & Coalesce

Coalesce - It can only decrease the partition,

Less shuffle will be happen

size of partitions will not be same

it is faster than the repartition because it avoid full shuffle of data

repartition - This method is use to increase or decrease the number of partitions

more shuffle will happen

It will create approximately equal size of partitions

4. Spark - Why can't we increase partition in coalesce?

- coalesce method did not create the new partitions to reduce them

It use the existing partitions to reduce and storing the data

5. Spark – What are the use cases for repartition & Coalesce

- If we want to increase the number of partitions and need equal size of the partitions then use repartition

If we want to only decrease the number of partitions and ok with the different size of the partition go with coalesce

6. Spark - What is better RDD or Dataframes and why?

- Dataframe is better option than RDD, because the RDD API is slower while grouping and aggregation

Dataframe APi are easy to use and faster as compare to RDD

7. Spark - What optimizer is used for Spark Dataframe?

- Spark use Catalyst Optimizer

8. Spark - How catalyst optimizer works in Spark?

The catalyst optimizer work in four stages

1 - analysis - here the unresolved plan get converted into the logical plan

2 - logical optimization - in this stage the logical plan get converted into the optimizes logical plan

3 - physical planning - the optimizes logical plan get converted into the physical plan

it went through the cost model and most optimized physical plan get selected

4 - code generation - here RDD is generated from the selected physical plan

9. Spark - What are the broadcast and accumulator variables?

- broadcast variables - It allow us to cached the read only variable on each machine, no need to send the variable on each machine

Spark also try to distribute the variable using efficient broadcast algorithms to reduce communication cost

- accumulator variables - This variables are used for the aggregating the information across the executors, this variable have write only permission at executors and read only permission on driver

10. Spark – What is Virtual partition and physical partition with respect to spark?

11. Spark - How do you decide the number of partitions in shuffle operation?

The number of partitions in shuffle operation should be equal to or in multiple of the number of core per executors, If we have 5 core on executor then 5 task can be run parallelly on that executor, the partition should be in multiple of that like 5, 10, 15

12. Spark - Explain different windowing functions in spark

Spark windowing functions include

lead(), lag()

Ranking functions - row_number(), rank(), dense_rank()

aggregation functions like avg(), sum()

13. Spark – What are different parameters required for spark-submit command? Difference between executors and cores? What is shuffle partition?

--deploy-mode - client , cluster

--master - yarn, meson standalone, kubernetes

--driver-memory -memory to be used by driver

--executor-memory -Amount of memory used by executor

--executor-cores -Number of CPU core use to be use for the executor

- -total-executor-core -The total number of executor cores to use

–conf - Use to specify the configuration properties of spark

Executors - are responsible to execute the task on the worker nodes

we can calculate the number of executors by (total number of core/ core per executor)

cores - The core are responsible to run the task in parallel, the value to core will decide the number of task can run in parallel

What is shuffle partition?

- when we perform the transformation on the dataframe which include the shuffling of data

then the new partition will be created, The number of the partitions will depends on the "spark.sql.shuffle.partitions" default value is 200 we can change it by spark.conf.set("spark.sql.shuffle.partitions", "200")

Hive

13. Hive – What are different types of partitioning mechanisms? Difference between them? Use cases of them?

There are two type of partitioning in hive

1- Static Partitioning

2- Dynamic Partitioning

Static Partitioning-1 we need to Insert input data files into the partition individually

2- when we load the data into hive table the static partitioning preferred

3- we can alter the partition in static partition

4 - we need to use where clause

Dynamic Partitioning - 1- we use single insert statement to insert the data into table

2- It take more time compare to static partitioning

3- we cannot perform the alter on dynamic partitioning

4 - No need to use where clause while loading data

14. Hive - What are the performance tuning mechanisms?

a - Avoid locking of tables - Make sure the table we are using in the query are not used by any other process. The locking of table can happen and our query may stuck for sometime

b - Use the Hive execution engine as TEZ - It is the preferred choice while executing hive query because it eliminates the unnecessary disk access . It take data from disk once, perform operations and produce output save us from multiple disk access

c - Use Hive Cost Based Optimizer (CBO) - Apache hive provide the cost based optamiser

to improve the performance. It generate the efficient execution plan like the order of join which type of join to perform etc by examine the query

d - Select columns which are needed -

Avoid using SELECT * FROM when we need some specific columns, Selecting all columns added unnecessary time

e - Suitable Input format Selection -

f - (Filter) the data as early as possible - This can avoid the long running of queries

for example -

a join b where a.col != null

above can be written as

(select * from a where a.col != null) join b

15. Hive - Use cases for Partition & Bucketing

Partition is a way to organize large tables into smaller logical tables based on values of columns In Hive, tables are created as a directory on HDFS. A table can have one or more partitions that correspond to a sub-directory for each partition inside a table directory.

number of partition will be created depend on the distinct value of the column

partition creates a directory having too many partitions creates too many sub-directories in a table directory

if we have a column having limited distinct value we can use partition , Partition column be decided on which we want to filter the data most

Bucketing - Hive Bucketing is a technique to split the data into more manageable files. You can also create bucketing on a partitioned table to further split the data which further improves the query performance of the partitioned table.

we define number of bucket should be created Each bucket is stored as a file within the table's directory or the partitions directories

If we have only unique data in the table we cannot use partition there because it will create partition for each record, here we should use bucketing

16. Hive - Upsert queries for parquet file

17. Hive - Create an external table in hive by referring schema of an existing table.

```
CREATE EXTERNAL  TABLE ext_table
LOCATION '/user/XXXXX/XXXXXX'
AS SELECT * from existing_table;
```

18. Hive - What is the purpose of msck repair in hive.

msck - metastore consistency check

lets say after creating hive table there is a change in HDFS data

after that when we query the hive table we will get the error

the meta data of the table is store in the metastore when we make some changes in the HDFS location directly the metastore not get updated

we have to sync the data with modified HDFS data

we can do this manually using msck repair command