NOTES ON APACHE HIVE - OPTIMIZATION TECHNIQUES

PART 01

linkedin.com/vaishnavi-muralidhar

HIVE OPTIMIZATION

Hive Optimization Theory:
There are 3 main ways in which optimization is achieved in Hive:

1. **Table structure level optimization**

-->Optimization achieved during the design of table structure itself.
There are 2 sub-processes under it:
1.Partitioning
2.Bucketing

2. **Query level optimization**

-->Optimization achieved while writing a query and reducing its run time
-->"Joins" are those which take more time for execution
-->we will be mostly focus on "join level optimization

3. **Simplifying query expressions to be written**

-->we will look on ways where we can simplify/reduce the code to be written
-->we will focus mainly on "window functions"

Sampling in Hive:
-->It is the process of generating a random/selected sample of data from the hive table which might be used for analyzing the data.

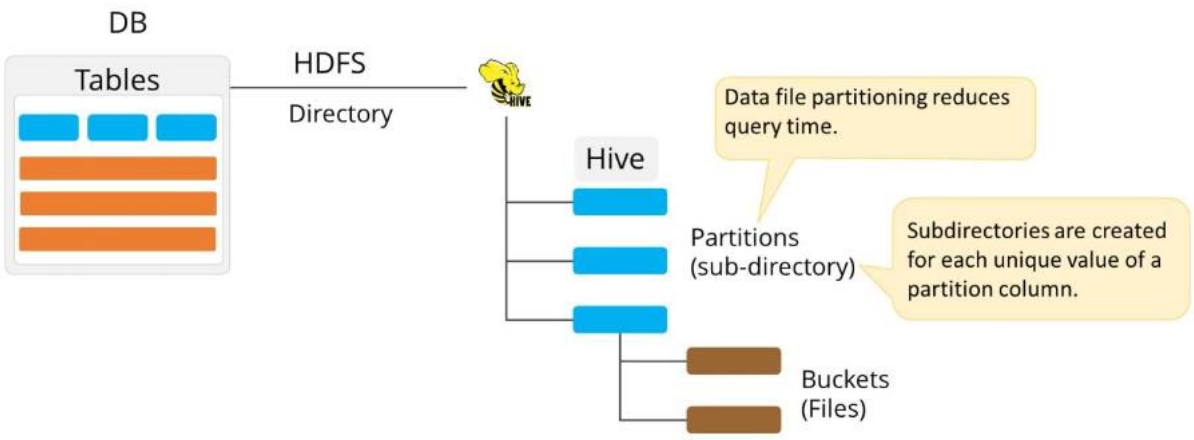
Few methods to do it :

1. Random Selection
2. Bucketized tables

PARTITIONING IN HIVE:
How to divide the data based on logical partitions?

-->Based on the table level optimization, one of the method to achieve it is through partitioning.
-->Here we divide the whole data into smaller chunks based on column values which are then partitioned into multiple directories
-->Due to this, the amount of data to be processed to give results will decrease resulting in fast retrieval of data.

Consider records are spread across multiple locations.
select * from orders
-->This will result in scanning the data records across all locations which consumes huge in-memory for processing



-->If we have logically partitioned the data based on location, then data records of particular location are scanned and optimizes the resources which is achieved through the concept of partitioning.
select * from orders where city ='BLR'

Directory in which table data stored:
--> /user/hive/warehouse/test.db/orders

Directory in which parition data is stored:
--> /user/hive/warehouse/test.db/orders/city="delhi"
/user/hive/warehouse/test.db/orders/city="hyd"

WHY Partitioning?
-->For faster retrieval of data records
-->To reduce the computation capacity

WHEN Partitioning?
-->when we have huge data which is stand alone
-->when we have logical column values based on which data can be partitioned
-->If the cardinality is low ie no. of distinct values is low then we should go with partitions.
No. of partitions = No. of distinct values in the column

Types of Partitioning:

-->There are 2 types of partitioning namely

3. **static partitioning**

-->In static partitioning we need to have an idea about the data and we have to load each of the partitions and mention them manually which is a tiring process.
-->static partitioning is faster than dynamic partitioning as we load the partitioned data rather than asking the machine to partition data for us.

Steps to follow:

4. create a empty table where u don't mention the partition column and load hdfs data into it

create table orders_NO_partition(order_id string,cust_id string,id string,quantity int,amount float,zipcode char(6), state char(2))

5. create a partition table

create table orders_partition(order_id string,cust_id string,id string,quantity int,amount float,zipcode char(6))
partitioned by (state char(2)) //we don't mention this column in table as field as data is partitioned based on this and causes data redundancy.
row format delimited
fields terminated by ','

6. Load each partition separately into the table

load data local inpath '/home/cloudera/Desktop/shared/week5_datasets/orders_CT_with_state.csv' into table orders_partition partition (**state='CT'**);
load data local inpath '/home/cloudera/Desktop/shared/week5_datasets/orders_CA_with_state.csv' into table orders_partition partition (**state='CA'**);

-->As you can observe **we have to load each partition separately** which is not possible in real world scenario where we don't have much idea about data

```
    > show partitions orders_partition;
OK
state=CA
state=CT
Time taken: 0.442 seconds, Fetched: 2 row(s)
hive>
```

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/orders_partition
Found 2 items
drwxrwxrwx   - cloudera supergroup          0 2021-07-02 11:02 /user/hive/warehouse/orders_partition/state=CA
drwxrwxrwx   - cloudera supergroup          0 2021-07-02 11:06 /user/hive/warehouse/orders_partition/state=CT
[cloudera@quickstart ~]$
```

7. **Dynamic partitioning**

-->In dynamic partitioning, the partitions are created automatically without manual intervention and respective directories are created. it

takes more time to do s machine has to be decide on how the partitions have to be created

Steps to follow:

8. Set properties to True and create a table without partitions and load data into it
9. create a partitions table and load above table data into it
10. now check for partitions that are dynamically created

-->Partitions are created by system dynamically at run time

```
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1625246213980_0001, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1625246213980
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1625246213980_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2021-07-02 11:44:06,451 Stage-1 map = 0%,  reduce = 0%
2021-07-02 11:44:29,907 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.43 sec
MapReduce Total cumulative CPU time: 2 seconds 430 msec
Ended Job = job_1625246213980_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/orders_dynamic_partition1/.hive-staging_hive_2021
Loading data to table default.orders_dynamic_partition1 partition (state=null)
         Time taken for load dynamic partitions : 3740
        Loading partition {state=CA}
        Loading partition {state=CT}
        Loading partition {state=NY}
         Time taken for adding to write entity : 163
Partition default.orders_dynamic_partition1{state=CA} stats: [numFiles=1, numRows=4, totalSize=104, rawDataSize=100]
Partition default.orders_dynamic_partition1{state=CT} stats: [numFiles=1, numRows=4, totalSize=120, rawDataSize=116]
Partition default.orders_dynamic_partition1{state=NY} stats: [numFiles=1, numRows=4, totalSize=128, rawDataSize=124]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 2.43 sec   HDFS Read: 5821 HDFS Write: 575 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 430 msec
OK
Time taken: 56.325 seconds
hive>
    >
    > select * from orders_dynamic_partition1;
OK
p1      c1      p1      NULL    1.11    90111   CA
p2      c2      p2      NULL    2.22    90222   CA
p3      c3      p3      NULL    3.33    90333   CA
p4      c4      p4      NULL    4.44    90444   CA
p10     c10     p10     NULL    10.11   90011   CT
p20     c20     p20     NULL    20.22   90022   CT
p30     c30     p30     NULL    30.33   90033   CT
p40     c40     p40     NULL    40.44   90044   CT
p100    c100    p10     NULL    10.11   90011   NY
p200    c200    p20     NULL    20.22   90022   NY
p300    c300    p30     NULL    30.33   90033   NY
p400    c400    p40     NULL    40.44   90044   NY
Time taken: 0.575 seconds, Fetched: 12 row(s)
hive>
    >
```

-->Partitions are created automatically

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/orders__dynamic_partition1
ls: `/user/hive/warehouse/orders__dynamic_partition1': No such file or directory
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/orders_dynamic_partition1
Found 3 items
drwxrwxrwx   - cloudera supergroup          0 2021-07-02 11:44 /user/hive/warehouse/orders_dynamic_partition1/state=CA
drwxrwxrwx   - cloudera supergroup          0 2021-07-02 11:44 /user/hive/warehouse/orders_dynamic_partition1/state=CT
drwxrwxrwx   - cloudera supergroup          0 2021-07-02 11:44 /user/hive/warehouse/orders_dynamic_partition1/state=NY
[cloudera@quickstart ~]$
```

**

-->If I don't create a main table and directly load external data into partitioned table. Only loading happens but not processing of partitions due to which we need to follow above procedure.

BUCKETING:

How to query data that is distinct and frequently queried?

-->when the data has high cardinality (no. of distinct values high) then we go for bucketing rather than partitioning.

-->Bucketing splits the data into smaller chunks based on a hash function of column value and stores the results in a file under table directory

-->No. of buckets have to be fixed during table creation. It has to be manually entered and we can decide on its value.

-->No. of buckets = No. of reducers (depends on type of operation invoked)

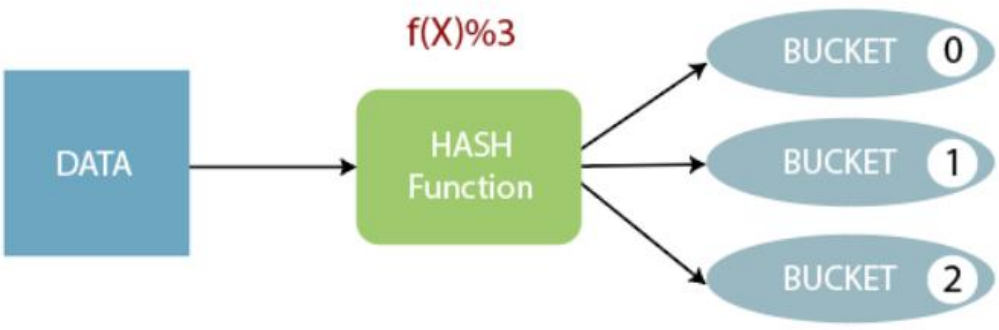
-->As each bucket undergoes hash function which performs shuffle & sort as in reducers so each bucket needs one reducer.

In HDFS:

-->Each partition is stored as folder

-->Each bucket is stored as file

In the below example, a simple hash function is applied on the product id's. Based on their values it is sent to respective buckets



-->Amount of data in a partition depends on the no. of records for that particular column
-->Amount of data in a bucket depends on a hash function due to which it is almost evenly distributed

WHY BUCKETING:
-->It gives faster query results
-->It optimizes join operations (if we know where our join condition(product_id) is present in a faster way then it will be easy to perform the join operation)

Steps to follow:
1.Create a table without buckets and load data from hdfs
2.Create a table with buckets
3.Transfer the data to no bucketed table to bucketed table

create table products_buckets(id int,name string,amount float,category string) **clustered by (id) into 4 buckets** row format delimited fields terminated
by ',' ;

-->we got 4 bucket files where our product_id is bucketed based on hash function
-->In file 1 ie hash function returned remainder 1 (9,5)

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/products_buckets
Found 4 items
-rwxrwxrwx   1 cloudera supergroup         23 2021-07-03 00:07 /user/hive/warehouse/products_buckets/000000_0
-rwxrwxrwx   1 cloudera supergroup         71 2021-07-03 00:07 /user/hive/warehouse/products_buckets/000001_0
-rwxrwxrwx   1 cloudera supergroup         60 2021-07-03 00:07 /user/hive/warehouse/products_buckets/000002_0
-rwxrwxrwx   1 cloudera supergroup         71 2021-07-03 00:07 /user/hive/warehouse/products_buckets/000003_0
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -cat /user/hive/warehouse/products_buckets/000000_0
8,earings,69.0,fashion
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -cat /user/hive/warehouse/products_buckets/000000_1
cat: `/user/hive/warehouse/products_buckets/000000_1': No such file or directory
[cloudera@quickstart ~]$ hadoop fs -cat /user/hive/warehouse/products_buckets/000001_0
9,chair,129.0,furniture
5,Nokia Y,39.99,mobile
1,iPhone,379.99,mobiles
[cloudera@quickstart ~]$
```

Partitioning & Bucketing:

-->we can combine both partitioning and bucketing together for better optimization.

-->Partitions are further subdivided into buckets

-->how the data is stored in hive

-->/user/hive/warehouse/test.db/orders/city="delhi"

bucket_0_file

bucket_1_file

bucket_2_file

bucket_3_file

-->The above folder stores all the data records belonging to city ="delhi" along with product_id which are bucketed based on hash value

ie (product_id % 4)

-->Partition based on category and bucketing on product_id for those partitions

create table partition_bucket(order_id string,cust_id string,id string,quantity int,amount float,zipcode char(6),state char(2))

partitioned by (category string)

clustered by (id) into 4 buckets

row format delimited

fields terminated by ',' ;

insert into covid_india_opt **partition (state)**
select * from covid_india;

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/partition_with_buckets
Found 5 items
drwxrwxrwx   - cloudera supergroup          0 2021-07-03 00:28 /user/hive/warehouse/partition_with_buckets/category=fashion
drwxrwxrwx   - cloudera supergroup          0 2021-07-03 00:28 /user/hive/warehouse/partition_with_buckets/category=furniture
drwxrwxrwx   - cloudera supergroup          0 2021-07-03 00:28 /user/hive/warehouse/partition_with_buckets/category=mobile
drwxrwxrwx   - cloudera supergroup          0 2021-07-03 00:28 /user/hive/warehouse/partition_with_buckets/category=mobiles
drwxrwxrwx   - cloudera supergroup          0 2021-07-03 00:28 /user/hive/warehouse/partition_with_buckets/category=toys
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/partition_with_buckets/category=fashion
Found 4 items
-rwxrwxrwx   1 cloudera supergroup         15 2021-07-03 00:28 /user/hive/warehouse/partition_with_buckets/category=fashion/000000_0
-rwxrwxrwx   1 cloudera supergroup          0 2021-07-03 00:28 /user/hive/warehouse/partition_with_buckets/category=fashion/000001_0
-rwxrwxrwx   1 cloudera supergroup          0 2021-07-03 00:28 /user/hive/warehouse/partition_with_buckets/category=fashion/000002_0
-rwxrwxrwx   1 cloudera supergroup         15 2021-07-03 00:28 /user/hive/warehouse/partition_with_buckets/category=fashion/000003_0
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -cat /user/hive/warehouse/partition_with_buckets/category=fashion/*
8,earings,69.0
7,makeup,100.0
[cloudera@quickstart ~]$
```