**Hive**

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Hive provides an SQL dialect, called Hive Query Language (abbreviated HiveQL or just HQL) for querying data stored in a Hadoop cluster. Hive is most suited for data warehouse applications, where relatively static data is analyzed, fast response times are not required, and when the data is not changing rapidly.

Performance optimization in hive:

1. Partitioning and bucketing
2. Indexes
3. Vectorization
4. Tez engine

2 execution engines- MR and tez. To check for engine set hive.execution.engine;

Set it to set hive.execution.engine=tez;

1. Bucket map join
2. Bucket map sorted merge join

Commonly used commands in hive :

1. Create table

create external table result(id int, name string, score int, dept string)

row format delimited

fields terminated by ','

stored as textfile

location '/home/result' ;

1. Load data from local csv files

load data local inpath '/home/result1/res.csv' overwrite into table result

1. There are 2 types of partitions in hive tables – static and dynamic

Static partition

create external table resultstatic(id int,name string,score int)

PARTITIONED by (dept string)

row format delimited fields terminated by ','

stored as textfile location '/home/result/static' ;

1. Load data in static partition from another hive table

insert overwrite table resultstatic partition (dept ='CS') select id, name, score from result where dept = 'CS';

insert overwrite table result partition (dept ='IT') select id,name,score from res where dept = 'IT';

1. Dynamic partition

Properties to set for dynamic partitioning

**set hive.exec.dynamic.partition = true;**

**set hive.exec.dynamic.partition.mode = nonstrict;**

create external table resultdyna1(id int,name string,score int)

PARTITIONED by (dept string)

row format delimited

fields terminated by ','

stored as textfile

location '/home/result/dynamic' ;

1. Load data in dynamic partitioned table

insert overwrite table resultdyna1 partition (dept) select id, name, score, dept from result;

1. Bucketing

When we write data in bucketed table in hive, it places the data in distinct buckets as files. Hive uses some hashing algorithm to generate a number in range of 1 to N buckets and based on the result of hashing, data is placed in a particular buckets as a file

**set hive.exec.dynamic.partition = true;**

**set hive.exec.dynamic.partition.mode = nonstrict;**

**set hive.enforce.bucketing=true;**

create external table result\_buck(id int,name string,score int)

PARTITIONED by (dept string)

clustered by (id) into 2 buckets

row format delimited

fields terminated by ','

stored as textfile

location '/home/result/bucket' ;

1. Load data in bucketed table

insert overwrite table result\_buck partition (dept) select id,name,score,dept from result;

1. Complex data types in hive

Array, Map, Struct

CREATE TABLE employees (

name STRING,salary FLOAT,subordinates ARRAY<STRING>,

deductions MAP<STRING, FLOAT>,address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>);

1. Map side join

Joins between two tables are performed in the Map phase

**set hive.auto.convert.join = true;**

SELECT /\*+ MAPJOIN(dataset2) \*/ dataset1.first\_name, dataset1.id,dataset2.id FROM dataset1 JOIN dataset2 ON dataset1.first\_name = dataset2.first\_name;

1. Bucket Map join

If tables being joined are bucketed on the joined columns, the number of buckets of one table is the multiple of bucket number of other table, the buckets can be joined together.

**Set hive.optimize.bucketmapjoin = true;**

SELECT /\*+ MAPJOIN(dataset2\_bucketed) \*/ dataset1\_bucketed.first\_name,dataset1\_bucketed.eid, dataset2\_bucketed.eid FROM dataset1\_bucketed JOIN dataset2\_bucketed ON dataset1\_bucketed.first\_name = dataset2\_bucketed.first\_name ;

1. Case statement

Select course, dept,

Case when sum (case when status in (‘fail’,’detained’) then 1 else 0 end) > 0 then ‘fail’

When sum (case when status in (‘absent’) then 1 else 0 end) > 0 then ‘absent’

When sum (case when status in (‘pass’) then 1 else 0 end) > 0 then ‘pass’

Else ‘no result’

End as final\_status

From college group by course, dept;

1. Skew join solution

Create 2 queries. 1 with the skewjoin key and another without the key

Do two separate queries

* select A.id from A join B on A.id = B.id where A.id <> 1;
* select A.id from A join B on A.id = B.id where A.id = 1 and B.id = 1;

Set the below properties

Set hive.optimize.skewjoin = true

Set hive.skewjoin.key = 10000

Set hive.skewjoin.mapjoin.map.tasks = 1000

Set hive.skewjoin.mapjoin.min.split = 33554432

1. Hiveconf and hivevar– to send variables dynamically

e.g. HQL file sample.hql

Select \* from ${hiveconf:table} where id=${hiveconf:id};

Set variable in shell script

Tablename=emp

Id=1

Set variable using hiveconf

hive -f "/home/sudha\_kelkar1990/sample.hql" --hiveconf table=$tablename --hiveconf id=$id;

1. Functions in hive

Standard functions – UDFs ( User Defined Functions)

UDFs are a powerful feature that allow users to extend HiveQL.

/\*\* A simple UDF to convert Celcius to Fahrenheit \*/

public class ConvertToCelcius extends UDF {

public double evaluate(double value) {

return (value - 32) / 1.8;

}

}

Once compiled, you can invoke an UDF like below:

hive> addjar my-udf.jar

hive> create temporary function fahrenheit\_to\_celcius using "com.mycomp.hive.udf.ConvertToCelcius";

hive> SELECT fahrenheit\_to\_celcius(temp\_fahrenheit) from temperature\_data;

1. Aggregate Functions

Another type of function is an aggregate function. All aggregate functions, user-defined

and built-in, are referred to generically as user-defined aggregate functions (UDAFs).

1. Table Generating Functions (UDTFs)

A third type of function supported by Hive is a table generating function.

hive> SELECT name, sub

> FROM employees

> LATERAL VIEW explode(subordinates) subView AS sub;

John Doe Mary Smith

John Doe Todd Jones

Mary Smith Bill King