**Hive Optimization Techniques:**

1. Partitions/Index/Buckets

2. Map Joins

Set hive.auto.convert.join=true;

3. Vectorization

Hive>set hive.vectorized.execution.enabled = true

5. File Formats(RC, ORC, PARQUET, AVRO)

6. Controlling no:of reducers

Hive> set hive.mapred.reduce.tasks=4;

7. Parallel execution

Hive>set hive.exec.parallel=true;

8. De-normalizing data

9. Final Output Compressions

SET hive.exec.compress.output=true;

SET mapred.output.compression.type=BLOCK;

SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCodec;

10. Intermediate Compression

hive.exec.compress.intermediate=true;

If a Hive query has more than one map reduce job, the contents of the intermediate files between jobs can be compressed with the following setting in the hive-site.xml file.

11. CBO(Cost Based Optimization)

set hive.cbo.enable=true;

set hive.compute.query.using.stats=true;

**Condition is Table should be ORC File Format Table**

**12. Using TEZ Execution Engine**

**Set hive.execution.engine;**

In MR, There will be multiple reads & writes like input data has to be read from Disk, execute map task, Write data to disk, run shuffle & sort and write back to disk, run reduce tasks which will read sorted data from disk & write back the output in disk. So multiple reads & writes will make the processing slow. That is where Horton works invented tez engine (before spark) which reads data once apply the logics and write final output on disk. It will hold all the data temporarily in memory & there by query speed is fast.

in MR shuffle is performed the same way regardless of the data size. Sorted partitions are written to disk, pulled by the reducers, merge-sorted and then fed into the reducers. Tez allows for small datasets to be handled entirely in memory, while no such optimization is available in map-reduce. Many warehousing queries sort or aggregate small datasets after the heavy lifting is done. These would benefit from an in memory shuffle.

Tez follows DAG (Directed acyclic graph) architecture. A typical Map reduce job has following steps:

1. Read data from file -->1st disk access

2. Run mappers

3. Write map output --> 2nd disk access

4. Run shuffle and sort --> read map output, 3rd disk access

5. write shuffle and sort --> write sorted data for reducers --> 4th disk access

6. Run reducers which reads sorted data --> 5th disk output

7. Write reducers output -->6th disk access

Tez works very similar to Spark (Created by Hortonworks before Spark):

Once ready to do some calculations (similar to actions in spark), get the data from disk and perform all steps and produce output.

TEZ follows :::::**Only one read and one write.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Compressions in hive \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**SET hive.exec.compress.output=true;**

**SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCodec;**

**SET mapred.output.compression.type=BLOCK;**

**SET hive.exec.compress.output=true;**

**SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCodec;**

**SET mapred.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec;**

**SET mapred.output.compression.codec=org.apache.hadoop.io.compress.DefaultCodec;**

**SET mapred.output.compression.codec=org.apache.hadoop.io.compress.Bzip2Codec;**

**SET mapred.output.compression.codec=org.apache.hadoop.io.compress.lzo.LzopCodec;**

++++++++++++++++++Optimizations in PIG ++++++++++++++++++++++

1. Push Up Filters to avoid unwanted data & Add ForEach to remove unneeded columns

2. If there is a chance, avoid filter & prefer split by

3. LimitOptimizer - Limit as early as possible

4. PushDownJoin - Apply Join as late as possible

6. Go for Specialized Joins like Replicated, Skewed & Merge Joins

7. Multi-Query Execution ( By default it is ON. To turn it off, "-M or -no\_multiquery"

8. Incresing the no: reducers by using parallel keyword.

9. Error Handling

In some cases it might be desirable to fail the entire script upon detecting the first failed job.

This can be achieved with the "-F" or "-stop\_on\_failure" command line flag.

10. Prefer DISTINCT over GROUP BY/GENERATE

To extract unique values from a column in a relation you can use DISTINCT or GROUP BY/GENERATE.

DISTINCT is the preferred method; it is faster and more efficient

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SPECIALIZED JOINS in PIG \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1. REPLICATED :** The small data set is replicated across all the datanodes to reduce network traffic in the map side only. Indirectly join operation is getting done in map side.

A = LOAD 'big\_data.csv' AS (b1,b2,b3);

B = LOAD 'tiny\_data.csv' AS (t1,t2,t3);

C = join A by b1, B by t1 using 'replicated';

Note: The small file should be enough to fit in main memory.

**2. SKEWED :**

A = LOAD 'big\_data.txt' AS (b1,b2,b3); /\*Big Bag

B = LOAD 'massive\_data.txt' AS (m1,m2,m3); /\*Massive Bag

C = JOIN A BY b1, B BY m1 USING 'skewed';

If huge amt of data goes to a single reducer, then the data will be shared across few more reducers.

This type of joins are used when the data associated with a given key is too large to fit in memory.

In order to use Skewed join, It works only with 2 tables & not supported for more than 2 joins.

**3. MERGE JOIN:**

Merge Joins give you good performance if both the bags are sorted based on the common key.

C = JOIN emp BY deptno, dept BY deptno USING 'merge';

Conditions :

1. It will work only if the join keys are sorted in ascending order in both the bags.

2. It can be performed with only 2 bags

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COMPLEX DATATYPES IN PIG\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1. MAP Datatype**

cat samp.txt

Ramesh,[city#Bangalore],123

Arun,[city#Anantapur],345

Pranith,[city#US],456

A = load 'samp.txt' using PigStorage(',') as

(name:chararray,addr:map[],empno:int);

B = foreach A generate name,addr#'city',empno;

cat map.txt

[Name#John,Age#22]

[Name#Arun,Age#23]

[Name#Teju,Age#24]

[Name#Swetha,Age#25]

[Name#Ravi,Age#26]

Details = load '/user/cloudera/dvs/map.txt' as (col:map[]);

Names = foreach Details generate col#'Name' as Name,col#'Age' as Age;

Voters = filter Names by Age>=25;

store Voters into 'dvs\_hdfs/mapvoters' using PigStorage('|');

**2. TUPLE Datatype**

cat tuple.txt

(3,8,9)

(1,4,7)

(2,5,8)

A = load 'tup.txt' as (T:tuple(f1:int,f2:int,f3:int));

B = foreach A generate T.f1,T.f2;

**3. BAG Datatype**

cat bag.txt

{(3,8,9)}

{(1,4,7)}

{(2,5,8)}

A = load 'bag.txt' as (B:bag{tuple(f1:int, f2:int, f3:int)});

B = foreach A generate B.f1, B.f2;

**SORT BY**

Hive uses SORT BY to sort the rows based on the given columns per reducer. If there are more than one reducer, then the output per reducer will be sorted, but the order of total output is not guaranteed to be sorted.

If we set the number of reducers to 2, then the query using sort by on ‘salary‘ column will produce the following output:-

Hive> set mapred.reduce.tasks=2;

Hive> select \* from emp sort by sal;

7369 SMITH CLERK 800.0 NULL 20

7900 JAMES CLERK 950.0 NULL 30

7876 ADAMS CLERK 1100.0 NULL 20

7521 WARD SALESMAN 1250.0 500.0 30

7654 MARTIN SALESMAN 1250.0 1400.0 30

7844 TURNER SALESMAN 1500.0 0.0 30

7566 JONES MANAGER 2975.0 NULL 20

7788 SCOTT ANALYST 3000.0 NULL 20

7902 FORD ANALYST 3000.0 NULL 20

7934 MILLER CLERK 1300.0 NULL 10

7499 ALLEN SALESMAN 1600.0 300.0 30

7782 CLARK MANAGER 2450.0 NULL 10

7698 BLAKE MANAGER 2850.0 NULL 30

7839 KING PRESIDENT 5000.0 NULL 10

We can see that the rows in yellow are sorted and rows in Green are sorted among themselves. But the overall sorting order is not maintained. Both these set of rows were processed by different reducers. Sort by is used to sort the data in each reducer according to the user specified order.

**ORDER BY**

Order by guarantees the total ordering of the output. Even if there are multiple reducers, the overall order of the output is maintained.

Hive> select \* from emp order by sal;

The query using order by on ‘salary‘ column will produce the following output:-

7369 SMITH CLERK 800.0 NULL 20

7900 JAMES CLERK 950.0 NULL 30

7876 ADAMS CLERK 1100.0 NULL 20

7521 WARD SALESMAN 1250.0 500.0 30

7654 MARTIN SALESMAN 1250.0 1400.0 30

7934 MILLER CLERK 1300.0 NULL 10

7844 TURNER SALESMAN 1500.0 0.0 30

7499 ALLEN SALESMAN 1600.0 300.0 30

7782 CLARK MANAGER 2450.0 NULL 10

7698 BLAKE MANAGER 2850.0 NULL 30

7566 JONES MANAGER 2975.0 NULL 20

7902 FORD ANALYST 3000.0 NULL 20

7788 SCOTT ANALYST 3000.0 NULL 20

7839 KING PRESIDENT 5000.0 NULL 10

We can see that the overall order of sorting is maintained in the result by using Order By. Order by always takes 1 reducer even if we set multiple reducers because it guarantees global sorting of the data in the overall output.

## DISTRIBUTE BY:

Distribute By is used to distribute the rows to different reducers based on the value(s) of column(s). All rows with the same Distribute By columns will go to the same reducer. This is like partitioning in Map-Reduce, where all the records having same value of partition goes to the same reducer.

Hive> select \* from emp distribute by deptno;

7934 MILLER CLERK 1300.0 NULL 10

7566 JONES MANAGER 2975.0 NULL 20

7654 MARTIN SALESMAN 1250.0 1400.0 30

7698 BLAKE MANAGER 2850.0 NULL 30

7782 CLARK MANAGER 2450.0 NULL 10

7788 SCOTT ANALYST 3000.0 NULL 20

7844 TURNER SALESMAN 1500.0 0.0 30

7876 ADAMS CLERK 1100.0 NULL 20

7900 JAMES CLERK 950.0 NULL 30

7521 WARD SALESMAN 1250.0 500.0 30

7369 SMITH CLERK 800.0 NULL 20

7839 KING PRESIDENT 5000.0 NULL 10

7499 ALLEN SALESMAN 1600.0 300.0 30

All rows in Yellow (with even id) were sent to one reducer and all rows in Green (with Odd id)were went to second Reducer. Thus we can use distribute by to control the distribution of rows to different reducers.

**Distribute by with Sort by**

If we club Distribute by with Sort by, then we can control the clustering of rows inside each reducer based on the value of some columns.

Select \* from emp distribute by deptno, sort by sal;

7900 JAMES CLERK 950.0 NULL 30

7876 ADAMS CLERK 1100.0 NULL 20

7654 MARTIN SALESMAN 1250.0 1400.0 30

7844 TURNER SALESMAN 1500.0 0.0 30

7782 CLARK MANAGER 2450.0 NULL 10

7698 BLAKE MANAGER 2850.0 NULL 30

7566 JONES MANAGER 2975.0 NULL 20

7788 SCOTT ANALYST 3000.0 NULL 20

7369 SMITH CLERK 800.0 NULL 20

7521 WARD SALESMAN 1250.0 500.0 30

7499 ALLEN SALESMAN 1600.0 300.0 30

7839 KING PRESIDENT 5000.0 NULL 10

All rows in Yellow (with even id) were sent to one reducer and all rows in Green (with Odd id)were went to second Reducer. Inside each reducer, the rows are sorted by salary.

## Cluster by :

Cluster by is same as ‘Distribute by and Sort by’. But Cluster by does the distribution and sorting on same columns. If we want to distribute by some columns and then sort by some other columns, then we should use ‘ distribute by with sort by’ instead of ‘cluster by’. We will see what a query with ‘cluster by id‘ returns.

Select \* from emp cluster by empno is same as

Select \* from emp distribute by empno, sort by empno;