

Tuesday, November 18, 2014

Understanding Hive joins in explain plan output

Hive is trying to embrace CBO(cost based optimizer) in latest versions, and Join is one major part of it. Understanding join best practices and use cases is one key factor of Hive performance tuning.

This article will explain each kind of join and also use explain plan output to show the difference.

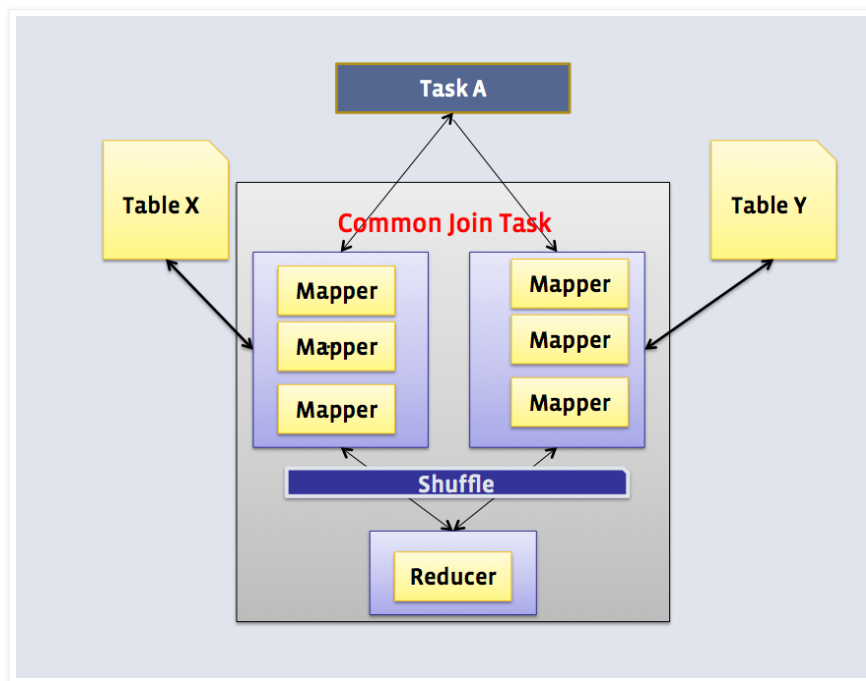
Note: All below tests are based on Hive 0.13.

1. Shuffle Join(Common Join).

How:

The shuffle join is the default option and it includes a map stage and a reduce stage.

- Mapper: reads the tables and output the join key-value pairs into an intermediate file.
- Shuffle: these pairs are sorted and merged.
- Reducer: gets the sorted data and does the join.



Use case:

It works for any table size.

Especially when other join types cannot be used, for example, full outer join.

Cons:

Most resource intensive since shuffle is an expensive operation.

Example:

```
hive> explain select a.* from passwords a, passwords2 b where a.col0=b.col1;
```

OK

STAGE DEPENDENCIES:

Stage-5 is a root stage , consists of Stage-1

Stage-1

Stage-0 is a root stage

STAGE PLANS:

Stage: Stage-5

Conditional Operator

Stage: Stage-1

Map Reduce

Map Operator Tree:

TableScan

alias: b

Search in

Search here..

About Op

OpenKB is just my
and share knowle
It may not be acc
be exactly what y
Any questions, ple
openkbinfo@gmail.com

San Jose,US :

Thu, 04/01/2014

Beijing,China:

Thu, 04/01/2014

Labels

- [AWS](#) (4)
- [Blogger](#) (5)
- [C#](#) (1)
- [cheat sheet](#) (1)
- [Cloudera Mana](#)
- [CUDA](#) (3)
- [Docker](#) (7)
- [drill](#) (49)
- [GCP](#) (1)
- [Good Reading](#)
- [GPU](#) (2)
- [Greenplum](#) (14)
- [Hadoop](#) (41)
- [HBase](#) (18)
- [hive](#) (42)
- [httpfs](#) (2)

```

Statistics: Num rows: 9961472 Data size: 477102080 Basic stats: COMPLETE Column stats: NONE
Reduce Output Operator
  key expressions: col1 (type: string)
  sort order: +
  Map-reduce partition columns: col1 (type: string)
  Statistics: Num rows: 9961472 Data size: 477102080 Basic stats: COMPLETE Column stats: NONE
  value expressions: col1 (type: string)

```

TableScan

```

alias: a
Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE
Reduce Output Operator
  key expressions: col0 (type: string)
  sort order: +
  Map-reduce partition columns: col0 (type: string)
  Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE
  value expressions: col0 (type: string), col1 (type: string), col2 (type: string), col3 (type: string), col4 (type: string), col5

```

Reduce Operator Tree:

Join Operator

```

condition map:
  Inner Join 0 to 1
condition expressions:
  0 {VALUE._col0} {VALUE._col1} {VALUE._col2} {VALUE._col3} {VALUE._col4} {VALUE._col5} {VALUE._col6}
  1 {VALUE._col1}
outputColumnNames: _col0, _col1, _col2, _col3, _col4, _col5, _col6, _col10
Statistics: Num rows: 10960295 Data size: 524940416 Basic stats: COMPLETE Column stats: NONE
Filter Operator
  predicate: (_col0 = _col10) (type: boolean)
  Statistics: Num rows: 5480147 Data size: 262470184 Basic stats: COMPLETE Column stats: NONE
Select Operator
  expressions: _col0 (type: string), _col1 (type: string), _col2 (type: string), _col3 (type: string), _col4 (type: string), _col5
  outputColumnNames: _col0, _col1, _col2, _col3, _col4, _col5, _col6
  Statistics: Num rows: 5480147 Data size: 262470184 Basic stats: COMPLETE Column stats: NONE
File Output Operator
  compressed: false
  Statistics: Num rows: 5480147 Data size: 262470184 Basic stats: COMPLETE Column stats: NONE
  table:
    input format: org.apache.hadoop.mapred.TextInputFormat
    output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
    serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

```

```

Stage: Stage-0
Fetch Operator
  limit: -1

```

Time taken: 1.707 seconds, Fetched: 58 row(s)

Tips:

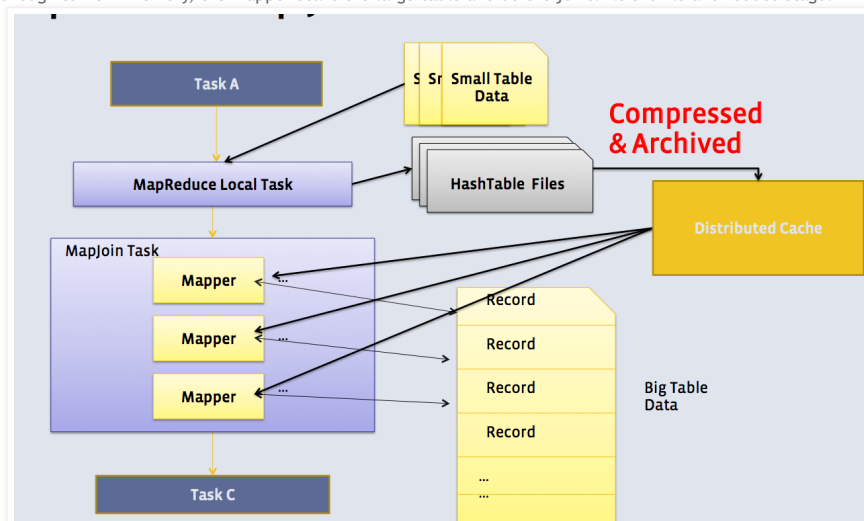
The largest table should be put on the rightmost since it should be the stream table.
However you can use hint "STREAMTABLE" to change the stream table in each map-reduce stage.

```
select /*+ STREAMTABLE(a) */ a.* from passwords a, passwords2 b, passwords3 c where a.col0=b.col0 and b.col0=c.col0;
```

2. Map Join(Broadcast Join)

How:

If one or more tables are small enough to fit in memory, the mapper scans the large table and do the joins. No shuffle and reduce stage.



Use case:

Small table(dimension table) joins big table(fact table). It is very fast since it saves shuffle and reduce stage.

Cons:

It requires at least one table is small enough.
Right/Full outer join don't work.

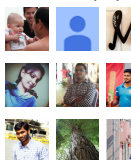
- hue (3)
- Impala (13)
- JAVA (25)
- Kafka (7)
- Kerberos (7)
- ksql (3)
- Kubernetes (12)
- MapR (44)
- MapR Stream (1)
- MapRDB (6)
- MySQL (2)
- Network (7)
- nvidia (4)
- Oozie (11)
- OS (20)
- parquet (7)
- pig (7)
- python (4)
- RAPIDS (12)
- scala (22)
- Spark (41)
- Spark Streaming (1)
- Tez (3)
- Tool (5)
- zookeeper (3)

Blog Arch

November 2014

Followers

Followers (93)



Follow

Example:

Here passwords3 table is very small table while passwords table is huge.

```
hive> explain select a.* from passwords a,passwords3 b where a.col0=b.col0;
OK
```

STAGE DEPENDENCIES:

```
Stage-4 is a root stage
Stage-3 depends on stages: Stage-4
Stage-0 is a root stage
```

STAGE PLANS:

Stage: Stage-4

Map Reduce Local Work

Alias -> Map Local Tables:

b

Fetch Operator

limit: -1

Alias -> Map Local Operator Tree:

b

TableScan

alias: b

Statistics: Num rows: 1 **Data size: 31** Basic stats: COMPLETE Column stats: NONE

HashTable Sink Operator

condition expressions:

0 {col0} {col1} {col2} {col3} {col4} {col5} {col6}

1 {col0}

keys:

0 col0 (type: string)

1 col0 (type: string)

Stage: Stage-3

Map Reduce

Map Operator Tree:

TableScan

alias: a

Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE

Map Join Operator

condition map:

Inner Join 0 to 1

condition expressions:

0 {col0} {col1} {col2} {col3} {col4} {col5} {col6}

1 {col0}

keys:

0 col0 (type: string)

1 col0 (type: string)

outputColumnNames: _col0, _col1, _col2, _col3, _col4, _col5, _col6, _col9

Statistics: Num rows: 10960295 Data size: 524940416 Basic stats: COMPLETE Column stats: NONE

Filter Operator

predicate: (_col0 = _col9) (type: boolean)

Statistics: Num rows: 5480147 Data size: 262470184 Basic stats: COMPLETE Column stats: NONE

Select Operator

expressions: _col0 (type: string), _col1 (type: string), _col2 (type: string), _col3 (type: string), _col4 (type: string), _c

outputColumnNames: _col0, _col1, _col2, _col3, _col4, _col5, _col6

Statistics: Num rows: 5480147 Data size: 262470184 Basic stats: COMPLETE Column stats: NONE

File Output Operator

compressed: false

Statistics: Num rows: 5480147 Data size: 262470184 Basic stats: COMPLETE Column stats: NONE

table:

input format: org.apache.hadoop.mapred.TextInputFormat

output format: org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat

serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

Local Work:

Map Reduce Local Work

Stage: Stage-0

Fetch Operator

limit: -1

Time taken: 0.1 seconds, Fetched: 63 row(s)

Tips:

1. Auto convert shuffle/common join to map join.

3 parameters are related:

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

```
set hive.auto.convert.join.noconditionaltask=true;
```

```
set hive.auto.convert.join.noconditionaltask.size=1000000;
```

Starting from Hive 0.11, hive.auto.convert.join=true by default.

You can disable this feature by setting hive.auto.convert.join=false.

When hive.auto.convert.join.noconditionaltask=true, if estimated size of small table(s) is smaller than hive.auto.convert.join.noconditionaltask.size(default 10MB), then common join can convert to map join automatically.

From above SQL plan output, we know estimated "Table b's Data Size=31" according to statistics.
If "set hive.auto.convert.join.noconditionaltask.size = 32;", the explain output shows map join operator:

Map Join Operator

If "set hive.auto.convert.join.noconditionaltask.size = 31;", then the join becomes common join operator:

Join Operator

2. Hint "MAPJOIN" can be used to force to use map join.

Before using the hint, firstly make sure below parameter is set to false(Default is true in Hive 0.13).

```
set hive.ignore.mapjoin.hint=false;
```

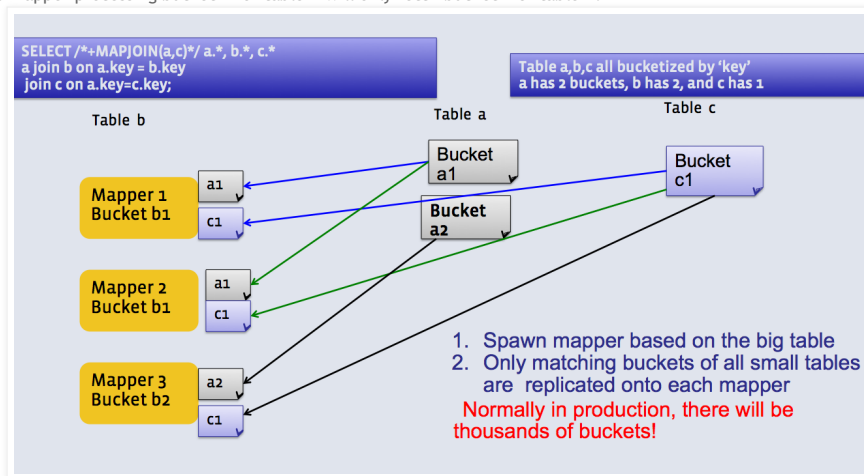
Then:

```
select /*+ MAPJOIN(a) */ a.* from passwords a, passwords2 b where a.col0=b.col0 ;
```

3. Bucket Map Join

How:

Join is done in Mapper only. The mapper processing bucket 1 for table A will only fetch bucket 1 of table B.



Use case:

When all tables are:

- Large.
- Bucketed using the join columns.
- The number of buckets in one table is a multiple of the number of buckets in the other table.
- Not sorted.

Cons:

Tables need to be bucketed in the same way how the SQL joins, so it cannot be used for other types of SQLs.

Tips:

1. The tables need to be created bucketed on the same join columns and also data need to be bucketed when inserting.

One way is to set "hive.enforce.bucketing=true" before inserting data.

For example:

```
create table b1(col0 string,col1 string,col2 string,col3 string,col4 string,col5 string,col6 string)  
clustered by (col0) into 32 buckets;  
create table b2(col0 string,col1 string,col2 string,col3 string,col4 string,col5 string,col6 string)  
clustered by (col0) into 8 buckets;
```

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

```
From passwords insert OVERWRITE table b1 select * limit 10000;
```

```
From passwords insert OVERWRITE table b2 select * limit 10000;
```

2. hive.optimize.bucketmapjoin must be set to true.

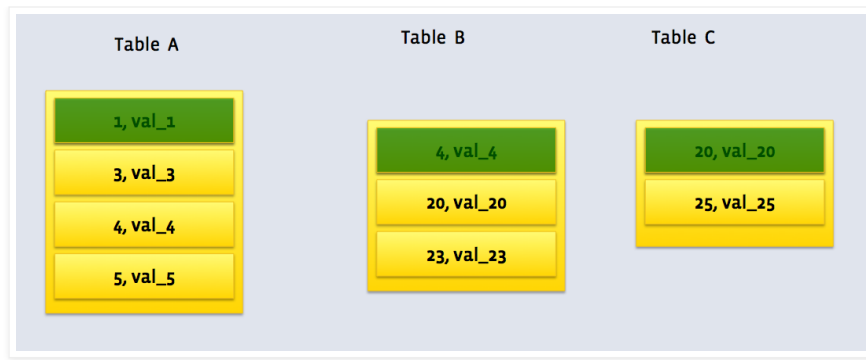
```
set hive.optimize.bucketmapjoin=true;
```

```
select /*+ MAPJOIN(b2) */ b1.* from b1,b2 where b1.col0=b2.col0 ;
```

4. Sort Merge Bucket(SMB) Map Join

How:

Join is done in Mapper only. The corresponding buckets are joined with each other at the mapper.



Use case:

When all tables are:

- Large.
- Bucketed using the join columns.
- Sorted using the join columns.
- All tables have the same number of buckets.

Cons:

Tables need to be bucketed in the same way how the SQL joins, so it cannot be used for other types of SQLs.

Partition tables might slow down.

Example:

```
hive> explain select c1.* from c1,c2 where c1.col0=c2.col0;
```

OK

STAGE DEPENDENCIES:

Stage-1 is a root stage

Stage-0 is a root stage

STAGE PLANS:

Stage: Stage-1

Map Reduce

Map Operator Tree:

TableScan

alias: c1

Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE

Sorted Merge Bucket Map Join Operator

condition map:

Inner Join 0 to 1

condition expressions:

0 {col0} {col1} {col2} {col3} {col4} {col5} {col6}

1 {col0}

keys:

0 col0 (type: string)

1 col0 (type: string)

outputColumnNames: _col0, _col1, _col2, _col3, _col4, _col5, _col6, _col9

Filter Operator

predicate: (_col0 = _col9) (type: boolean)

Select Operator

expressions: _col0 (type: string), _col1 (type: string), _col2 (type: string), _col3 (type: string), _col4 (type: string), _c

outputColumnNames: _col0, _col1, _col2, _col3, _col4, _col5, _col6

File Output Operator

compressed: false

table:

input format: org.apache.hadoop.mapred.TextInputFormat

output format: org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat

serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

Stage: Stage-0

Fetch Operator

limit: -1

Time taken: 0.134 seconds, Fetched: 37 row(s)

Tips:

1. The tables need to be created bucketed and sorted on the same join columns and also data need to be bucketed when inserting.

One way is to set "hive.enforce.bucketing=true" before inserting data.

For example:

```
create table c1(col0 string,col1 string,col2 string,col3 string,col4 string,col5 string,col6 string)
```

```
clustered by (col0) sorted by (col0) into 32 buckets;
```

```
create table c2(col0 string,col1 string,col2 string,col3 string,col4 string,col5 string,col6 string)
```

```
clustered by (col0) sorted by (col0) into 32 buckets;
```

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

```
From passwords insert OVERWRITE table c1 select * order by col0;
```

```
From passwords insert OVERWRITE table c2 select * order by col0;
```

2. Below parameters need to set to convert SMB join to SMB map join.

```

set hive.auto.convert.sortmerge.join=true;
set hive.optimize.bucketmapjoin = true;
set hive.optimize.bucketmapjoin.sortedmerge = true;
set hive.auto.convert.sortmerge.join.noconditionaltask=true;

```

3. Big table selection policy parameter "hive.auto.convert.sortmerge.join.bigtable.selection.policy" determines which table is for only streaming. It has 3 values:

```

org.apache.hadoop.hive.q1.optimizer.AvgPartitionSizeBasedBigTableSelectorForAutoSMJ (default)
org.apache.hadoop.hive.q1.optimizer.LeftmostBigTableSelectorForAutoSMJ
org.apache.hadoop.hive.q1.optimizer.TableSizeBasedBigTableSelectorForAutoSMJ

```

4. Hint "MAPJOIN" can determine which table is small and should be loaded into memory.

5. Small tables are read on demand which means not holding small tables in memory.

6. Outer join is supported.

5. Skew Join

How:

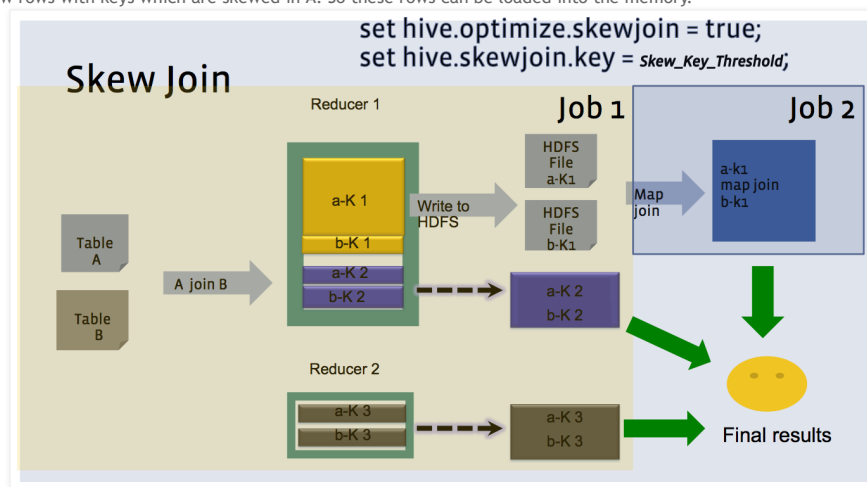
If table A join B, and A has skew data "1" in joining column.

First read B and store the rows with key 1 in an in-memory hash table. Now run a set of mappers to read A and do the following:

- If it has key 1, then use the hashed version of B to compute the result.
- For all other keys, send it to a reducer which does the join. This reducer will get rows of B also from a mapper.

This way, we end up reading only B twice. The skewed keys in A are only read and processed by the Mapper, and not sent to the reducer. The rest of the keys in A go through only a single Map/Reduce.

The assumption is that B has few rows with keys which are skewed in A. So these rows can be loaded into the memory.



Use case:

One table has huge skew values on the joining column.

Cons:

One table is read twice.

Users should be aware of the skew key.

Example:

```
hive> explain select a.* from passwords a, passwords2 b where a.col0=b.col1;
```

OK

STAGE DEPENDENCIES:

Stage-7 is a root stage , consists of Stage-1

Stage-1

Stage-4 depends on stages: Stage-1 , consists of Stage-8

Stage-8

Stage-3 depends on stages: Stage-8

Stage-0 is a root stage

STAGE PLANS:

Stage: Stage-7

Conditional Operator

Stage: Stage-1

Map Reduce

Map Operator Tree:

TableScan

alias: b

Statistics: Num rows: 9961472 Data size: 477102080 Basic stats: COMPLETE Column stats: NONE

Reduce Output Operator

key expressions: col1 (type: string)

sort order: +

Map-reduce partition columns: col1 (type: string)

Statistics: Num rows: 9961472 Data size: 477102080 Basic stats: COMPLETE Column stats: NONE

value expressions: col1 (type: string)

TableScan

alias: a

Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE

Reduce Output Operator

```
key expressions: col0 (type: string)
sort order: +
Map-reduce partition columns: col0 (type: string)
Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE
value expressions: col0 (type: string), col1 (type: string), col2 (type: string), col3 (type: string), col4 (type: string), col5
```

Reduce Operator Tree:

Join Operator

```
condition map:
  Inner Join 0 to 1
condition expressions:
  0 {VALUE._col0} {VALUE._col1} {VALUE._col2} {VALUE._col3} {VALUE._col4} {VALUE._col5} {VALUE._col6}
  1 {VALUE._col1}
```

handleSkewJoin: true

outputColumnNames: _col0, _col1, _col2, _col3, _col4, _col5, _col6, _col10

Statistics: Num rows: 10960295 Data size: 524940416 Basic stats: COMPLETE Column stats: NONE

Filter Operator

predicate: (_col0 = _col10) (type: boolean)

Statistics: Num rows: 5480147 Data size: 262470184 Basic stats: COMPLETE Column stats: NONE

Select Operator

expressions: _col0 (type: string), _col1 (type: string), _col2 (type: string), _col3 (type: string), _col4 (type: string), _col5

outputColumnNames: _col0, _col1, _col2, _col3, _col4, _col5, _col6

Statistics: Num rows: 5480147 Data size: 262470184 Basic stats: COMPLETE Column stats: NONE

File Output Operator

compressed: false

Statistics: Num rows: 5480147 Data size: 262470184 Basic stats: COMPLETE Column stats: NONE

table:

input format: org.apache.hadoop.mapred.TextInputFormat

output format: org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat

serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

Stage: Stage-4

Conditional Operator

Stage: Stage-8

Map Reduce Local Work

Alias -> Map Local Tables:

```
1
  Fetch Operator
  limit: -1
```

Alias -> Map Local Operator Tree:

```
1
  TableScan
  HashTable Sink Operator
  condition expressions:
    0 {0_VALUE_0} {0_VALUE_1} {0_VALUE_2} {0_VALUE_3} {0_VALUE_4} {0_VALUE_5} {0_VALUE_6}
    1 {1_VALUE_0}
  keys:
    0 joinkey0 (type: string)
    1 joinkey0 (type: string)
```

Stage: Stage-3

Map Reduce

Map Operator Tree:

TableScan

Map Join Operator

condition map:

Inner Join 0 to 1

condition expressions:

0 {0_VALUE_0} {0_VALUE_1} {0_VALUE_2} {0_VALUE_3} {0_VALUE_4} {0_VALUE_5} {0_VALUE_6}

1 {1_VALUE_0}

keys:

0 joinkey0 (type: string)

1 joinkey0 (type: string)

outputColumnNames: _col0, _col1, _col2, _col3, _col4, _col5, _col6, _col10

Filter Operator

predicate: (_col0 = _col10) (type: boolean)

Select Operator

expressions: _col0 (type: string), _col1 (type: string), _col2 (type: string), _col3 (type: string), _col4 (type: string), _c

outputColumnNames: _col0, _col1, _col2, _col3, _col4, _col5, _col6

File Output Operator

compressed: false

table:

input format: org.apache.hadoop.mapred.TextInputFormat

output format: org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat

serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

Local Work:

Map Reduce Local Work

Stage: Stage-0

Fetch Operator

limit: -1

Time taken: 0.331 seconds, Fetched: 110 row(s)

As above shows, there are 2 join operators, one is common join and the other one is map join.
And it shows "handleSkewJoin: true".

Tips:

- 1. Below parameter needs to be set to enable skew join.

```
set hive.optimize.skewjoin=true;
```

- 2. Below parameter determine if we get a skew key in join.
If we see more than the specified number of rows with the same key in join operator, we think the key as a skew join key.

```
set hive.skewjoin.key=100000;
```

===== Reference =====

- [Hive performance](#)
- [Join Optimization](#)
- [Skewed Join Optimization](#)
- [Joins](#)
- [Configuration Properties](#)
- [Join Strategies in Hive](#)


=====

Related Posts

"MscK Repair" Of A Partition Table Fails With Error "Expecting Only One Partition But More Than One Partitions Are Found."
Hive: Different Lock Behaviors Between DummyTxnManager And DbTxnManager
How To Install And Configure MapR Hive ODBC Driver On Linux
Error Java.Lang.NoSuchMethodException When Running Spark-Sql-Perf With Hive Metastore 3.X
Can Not Drop Partition Or Drop Hive Partition Table Due To UTF8 Characters In Partition Value
How To Use Snapshot Feature Of MapR-FS To Isolate Hive Reads And Writes

Posted by [OpenKB](#) at 7:26 PM
Labels: [hive](#)


5 comments:

- 

Unknown

September 28, 2017 at 9:51 AM

Thank you for the post! It was very useful for me.


[Reply](#)
- 

venky

October 31, 2017 at 10:34 AM

Its was very useful...Thank you for the post..


[Reply](#)

[Replies](#)
- 

OpenKB

October 31, 2017 at 1:38 PM

Good to know:)

[Reply](#)
- 

Unknown

January 4, 2018 at 1:47 PM

Thanks for the awesome post..

I'm using a simple query with 2 tables in from clause with a single inner join
1 of them is a large table and other one is a small table.
I'm trying to optimize the query by enforcing map join as mentioned here
When i enforce the parameters mentioned in your blog as mentioned, My run time is going higher than actual existing query. Can you please let me know how i can optimize my query and reduce the run time..

Without map join, my query run time is 38 seconds
With map join my query run time is 50 seconds
I would like to bring it down < 20s Can you please suggest any solution to this

[Reply](#)

[Replies](#)



OpenKB

January 4, 2018 at 1:59 PM

To compare the performance for below 2 scenarios:

1. Without map join
VS
2. With map join

You need to check the complete query log to determine:

- a. How many "Stages" are spawn for #1 and #2?
- b. Which "Stage" is causing the performance difference?
- c. For that problematic "Stage" in #2, I think it should be a map-reduce job, then you should check how many mappers and reducers are spawn for this map-reduce job. How about #1?

[Reply](#)

Enter your comment...



Comment as:

slidebasis88@ç ▾

[Sign out](#)

[Publish](#)

[Preview](#)

☐ **Notify me**

[Prev Page](#)

[Home](#)

[Next Page](#)

Subscribe to: [Post Comments \(Atom\)](#)

Popular Posts

[How to check JAVA memory usage](#)

Many commands can check the memory utilization of JAVA processes, for example, pmap, ps, jmap, jstat. What are the differences? Before we ...

[How to control the file numbers of hive table after inserting data on MapR-FS.](#)

Hive table contains files in HDFS, if one table or one partition has too many small files, the HiveQL performance may be impacted. Sometime...

[Understanding Hive joins in explain plan output](#)

Hive is trying to embrace CBO(cost based optimizer) in latest versions, and Join is one major part of it. Understanding join best practices ...

[Scala on Spark cheatsheet](#)

This is a cookbook for scala programming. 1. Define a object with main function -- Helloworld. object HelloWorld { def main(args: Array...

[How to use Scala on Spark to load data into Hbase/MapRDB -- normal load or bulk load.](#)

This article shows a sample code to load data into Hbase or MapRDB(M7) using Scala on Spark. I will introduce 2 ways, one is normal load us...

[How to build and use parquet-tools to read parquet files](#)

Goal: How to build and use parquet-tools to read parquet files. Solution: 1. Download and Install maven. Follow below link: http://...

[Memory allocation for Oozie Launcher job](#)

Goal: This article explains the configuration parameters for Oozie Launcher job.

[Difference between Spark HiveContext and SQLContext](#)

Goal: This article explains what is the difference between Spark HiveContext and SQLContext.

[How to list table or partition location from Hive Metastore](#)

Goal: This article provides the SQL to list table or partition locations from Hive Metastore. Env: Hive metastore 0.13 on MySQL Root ...

[Hive on Tez : How to control the number of Mappers and Reducers](#)

Goal: How to control the number of Mappers and Reducers in Hive on Tez.