Optimizing Hive queries for ORC formatted tables

SYNOPSIS

The Optimized Row Columnar (ORC) file is a columnar storage format for Hive. Specific Hive configuration settings for ORC formatted tables can improve query performance resulting in faster execution and reduced usage of computing resources. Some of these settings may already be turned on by default, whereas others require some educated guesswork.

The table below compares Tez job statistics for the same Hive query that was submitted without and with certain configuration settings. Notice the performance gains with optimization. This article will explain how the performance improvements were achieved.

	Optimized Query	Non-Optimized Query	% Improvement w/Optimized Query
ELAPSED TIME IN SECONDS	31.2	47.20	34%
TOTAL LAUNCHED TASKS	104	316	67%
HDFS BYTES READ	2,710,532,064	13,049,033,117	79%
HDFS READ OPERATIONS	178	599	70%
CPU MILLISECONDS	1,144,380	2,032,250	44%
PHYSICAL MEMORY IN BYTES	257,542,848,512	975,135,834,112	74%
GARBAGE COLLECTION TIME MILLISECONDS	35,768	43,306	17%

QUERY EXECUTION

Source Data:

- 102,602,110 Clickstream page view records across 5 days of data for multiple countries
- Table is partitioned by date in the format YYYY-MM-DD.
- There are no indexes and table is not bucketed.

The HiveQL is ranking each page per user by how many times the user viewed that page for a specific date and within the United States. Breakdown of the query:

1. Scan all the page views for each user.
2. Filter for page views on 1 date partition and only include traffic in the United States.
3. For each user, rank each page in terms of how many times it was viewed by that user.
4. For example, I view Page A 3 times and Page B once. Page A would rank 1 and Page B would rank 2.

Without optimization

```
Status: Running (Executing on YARN cluster with App id application_1480278870175_0021)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED     97         97        0        0       0       0
Reducer 2 ......   SUCCEEDED    146        146        0        0       0       0
Reducer 3 ......   SUCCEEDED     73         73        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 47.20 s
--------------------------------------------------------------------------------
Moving data to directory hdfs://ip-172-31-2-101.us-west-2.compute.internal:8020/apps/hive/warehouse/rank_page_views_orc
Table default.rank_page_views_orc stats: [numFiles=73, numRows=11045379, totalSize=920709323, rawDataSize=909663944]
OK
```

With optimization

```
Status: Running (Executing on YARN cluster with App id application_1480278870175_0022)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED     74         74        0        0       0       0
Reducer 2 ......   SUCCEEDED     20         20        0        0       0       0
Reducer 3 ......   SUCCEEDED     10         10        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 31.20 s
--------------------------------------------------------------------------------
Moving data to directory hdfs://ip-172-31-2-101.us-west-2.compute.internal:8020/apps/hive/warehouse/rank_page_views_orc
Table default.rank_page_views_orc stats: [numFiles=10, numRows=11045379, totalSize=920709323, rawDataSize=909663944]
OK
```

Notice the change in reducers

- The final output size of all the reducers is 920 MB.
- For the first run, 73 reducers completed resulting in 73 output files. This is excessive. 920 MB into 73 reducers is around 12.5 MB per reducer output. This is unnecessary overhead resulting in too many small files. More parallelism does not always equate to better performance.
- The second run launched 10 reducers resulting in 10 reduce files. 920 MB into 10 reducers is about 92 MB per reducer output. Much less overhead and we don't run into the small files problem. The maximum number of files in HDFS depends on the amount of memory available in the NameNode. Each block, file, and directory in HDFS is represented as an object in the NameNode's memory each of which occupies about 150 Bytes.

OPTIMIZATION

1. Always collect statistics on those tables for which data changes frequently. Schedule an automated ETL job to run at certain times:

ANALYZE TABLE page_views_orc COMPUTE STATISTICS FOR COLUMNS;

2. Run the Hive query with the following settings:

   SET hive.optimize.ppd=true;

   SET hive.optimize.ppd.storage=true;

   SET hive.vectorized.execution.enabled=true;

   SET hive.vectorized.execution.reduce.enabled = true;

   SET hive.cbo.enable=true;

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

   SET hive.stats.fetch.column.stats=true;

   SET hive.stats.fetch.partition.stats=true;

   SET hive.tez.auto.reducer.parallelism=true;

   SET hive.tez.max.partition.factor=20;

   SET hive.exec.reducers.bytes.per.reducer=128000000;

3. Partition your tables by date if you are storing a high volume of data per day. Table management becomes easier. You can easily drop partitions that are no longer needed or for which data has to be reprocessed.

SUMMARY

Let’s look at each of the Hive settings.

1. **Enable predicate pushdown (PPD) to filter at the storage layer:**

   SET hive.optimize.ppd=true;

   SET hive.optimize.ppd.storage=true

2. **Vectorized query execution processes data in batches of 1024 rows instead of one by one:**

   SET hive.vectorized.execution.enabled=true;

   SET hive.vectorized.execution.reduce.enabled=true;

3. **Enable the Cost Based Optimizer (COB) for efficient query execution based on cost and fetch table statistics:**

   SET hive.cbo.enable=true;

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

SET hive.stats.fetch.column.stats=true;

SET hive.stats.fetch.partition.stats=true;

Partition and column statistics from fetched from the metastsore. Use this with caution. If you have too many partitions and/or columns, this could degrade performance.

4. **Control reducer output:**

   SET hive.tez.auto.reducer.parallelism=true;

   SET hive.tez.max.partition.factor=20;

   SET hive.exec.reducers.bytes.per.reducer=128000000;

This last set is important. The first run produced 73 output files with each file being around 12.5 MB in size. This is inefficient as I explained earlier. With the above settings, we are basically telling Hive an approximate maximum number of reducers to run with the caveat that the size for each reduce output should be restricted to 128 MB. Let's examine this:

- The parameter hive.tez.max.partition.factor is telling Hive to launch up to 20 reducers. This is just a guess on my part and Hive will not necessarily enforce this. My job completed with only 10 reducers - 10 output files.
- Since I set a value of 128 MB for hive.exec.reducers.bytes.per.reducer, Hive will try to fit the reducer output into files that are come close to 128 MB each and not just run 20 reducers.
- If I did not set hive.exec.reducers.bytes.per.reducer, then Hive would have launched 20 reducers, because my query output would have allowed for this. I tested this and 20 reducers ran.
- 128 MB is an approximation for each reducer output when setting hive.exec.reducers.bytes.per.reducer. In this example the total size of the output files is 920 MB. Hive launched 10 reducers which is about 92 MB per reducer file. When I set this to 64 MB, then Hive launched the 20 reducers with each file being around 46 MB.
- If hive.exec.reducers.bytes.per.reducer is set to a very high value then you will have fewer reducers than if set to a lower value. Higher values result in fewer reducers being launched which can also degrade performance. You need just the right level of parallelism.