Hive Performance Tuning

7

This entry was posted in Hive *on May 3, 2015 by Siva*

In our previous post we have discussed about hadoop job optimization or Hadoop Performance Tuning for Mapreduce jobs. In this post we will briefly discuss a few points on how to optimize hive queries/ Hive Performance tuning.

If we do not fine tune Hive properly, then even for select queries on smaller tables in Hive, some times it may take minutes to emit results. So, because of this reason Hive is mainly limited to OLAP features only. When instant results expected then Hive is not suitable. But by following below practices we can improve the Hive query performances at least by 50 %.

Table of Contents [hide]

Hadoop Online Tutorial
A learning center for hadoop Eco System

Hive Performance Tuning:

Below are the list of practices that we can follow to optimize Hive Queries.

1. Enable Compression in Hive

By enabling compression at various phases (i.e. on final output, intermediate data), we achieve the performance improvement in Hive Queries. For further details on how to enable compression Hive refer the post Compression in Hive.

2. Optimize Joins

We can improve the performance of joins by enabling Auto Convert Map Joins and enabling optimization of skew joins.

Auto Map Joins

Auto Map-Join is a very useful feature when joining a big table with a small table. if we enable this feature, the small table will be saved in the local cache on each node, and then joined with the big table in the Map phase. Enabling Auto Map Join provides two advantages. First, loading a small table into cache will save read time on each data node. Second, it avoids *skew joins* in the Hive query, since the join operation has been already done in the Map phase for each block of data.

To enable the Auto Map-Join feature, we need to set below properties.

```
<property>
  <name>hive.auto.convert.join</name>
  <value>true</value>
  <description>Whether Hive enables the optimization about converting common join into ma
</property>
<property>
  <name>hive.auto.convert.join.noconditionaltask</name>
  <value>true</value>
  <description>
    Whether Hive enables the optimization about converting common join into mapjoin based
    If this parameter is on, and the sum of size for n-1 of the tables/partitions for a n
    specified size, the join is directly converted to a mapjoin (there is no conditional
  </description>
</property>
<property>
  <name>hive.auto.convert.join.noconditionaltask.size</name>
  <value>10000000</value>
  <description>
    If hive.auto.convert.join.noconditionaltask is off, this parameter does not take affe
    However, if it is on, and the sum of size for n-1 of the tables/partitions for a n-wa
    the join is directly converted to a mapjoin(there is no conditional task). The defau
  </description>
</property>
<property>
  <name>hive.auto.convert.join.use.nonstaged</name>
  <value>false</value>
  <description>
    For conditional joins, if input stream from a small alias can be directly applied to
    filtering or projection, the alias need not to be pre-staged in distributed cache via
    Currently, this is not working with vectorization or tez execution engine.
  </description>
</property>
```

Skew Joins

We can enable optimization of skew joins, i.e. imbalanced joins by setting **hive.optimize.skewjoin** property to **true** either via **SET** command in hive shell or **hive-site.xml** file. Below are the list of properties that can be fine tuned to better
ns.

```
<property>
  <name>hive.optimize.skewjoin</name>
  <value>true</value>
  <description>
    Whether to enable skew join optimization.
    The algorithm is as follows: At runtime, detect the keys with a large skew. Instead o
    processing those keys, store them temporarily in an HDFS directory. In a follow-up ma
    job, process those skewed keys. The same key need not be skewed for all the tables, a
    the follow-up map-reduce job (for the skewed keys) would be much faster, since it wou
    map-join.
  </description>
</property>
<property>
  <name>hive.skewjoin.key</name>
  <value>100000</value>
  <description>
    Determine if we get a skew key in join. If we see more than the specified number of r
    we think the key as a skew join key.
  </description>
</property>
<property>
  <name>hive.skewjoin.mapjoin.map.tasks</name>
  <value>10000</value>
  <description>
    Determine the number of map task used in the follow up map join job for a skew join.
    It should be used together with hive.skewjoin.mapjoin.min.split to perform a fine gr
  </description>
</property>
<property>
  <name>hive.skewjoin.mapjoin.min.split</name>
  <value>33554432</value>
  <description>
    Determine the number of map task at most used in the follow up map join job for a ske
    the minimum split size. It should be used together with hive.skewjoin.mapjoin.map.tas
  </description>
</property>
```

Enable Bucketed Map Joins

If tables are bucketed by a particular column and these tables are being used in joins then we can enable bucketed map join to improve the performance. To do this, we can set below properties in **hive-site.xml** or hive shell.

```
<property>
  <name>hive.optimize.bucketmapjoin</name>
  <value>true</value>
  <description>Whether to try bucket mapjoin</description>
</property>
<property>
  <name>hive.optimize.bucketmapjoin.sortedmerge</name>
  <value>true</value>
  <description>Whether to try sorted bucket merge map join</description>
</property>
```

3. Avoid Global Sorting in Hive

Global Sorting in Hive can be achieved in Hive with **ORDER BY** clause but this comes with a drawback. ORDER BY produces a result by setting the number of reducers to one, making it very inefficient for large datasets.

When a globally sorted result is not required, then we can use **SORT BY** clause. SORT BY produces a sorted file per reducer.

l which reducer a particular row goes to, we can use **DISTRIBUTE BY** clause, for example,

```
SELECT id, name, salary, dept FROM employee
DISTRIBUTE BY dept
SORT BY id ASC, name DESC;
```

Each ***dept*** will be processed separately by a reducer and records will be sorted by ***id*** and ***name*** fields within each ***dept*** separately.

4. Enable Tez Execution Engine

Instead of running Hive queries on venerable Map-reduce engine, we can improve the performance of hive queries at least by 100% to 300 % by running on Tez execution engine. We can enable the Tez engine with below property from hive shell.

```
hive> set hive.execution.engine=tez;
```

5. Optimize LIMIT operator

By default LIMIT operator still executes the entire query, then only returns a limited results. Because this behavior is generally wasteful, it can be avoided by setting below properties.

```
  <property>
    <name>hive.limit.optimize.enable</name>
    <value>true</value>
    <description>Whether to enable to optimization to trying a smaller subset of data for s
  </property>
  <property>
    <name>hive.limit.row.max.size</name>
    <value>100000</value>
    <description>When trying a smaller subset of data for simple LIMIT, how much size we ne
  </property>
  <property>
    <name>hive.limit.optimize.limit.file</name>
    <value>10</value>
    <description>When trying a smaller subset of data for simple LIMIT, maximum number of 
  </property>
  <property>
    <name>hive.limit.optimize.fetch.max</name>
    <value>50000</value>
    <description>
      Maximum number of rows allowed for a smaller subset of data for simple LIMIT, if it 
      Insert queries are not restricted by this limit.
    </description>
  </property>
```

6. Enable Parallel Execution

Hive converts a query into one or more stages. Stages could be a MapReduce stage, sampling stage, a merge stage, a limit stage. By default, Hive executes these stages one at a time. A particular job may consist of some stages that are not dependent on each other and could be executed in

parallel, possibly allowing the overall job to complete more quickly. Parallel execution can be enabled by setting below properties.

```
  <property>
    <name>hive.exec.parallel</name>
                ue>
                ther to execute jobs in parallel</description>
  </property>
  <property>
    <name>hive.exec.parallel.thread.number</name>
    <value>8</value>
    <description>How many jobs at most can be executed in parallel</description>
  </property>
```

7. Enable Mapreduce Strict Mode

we can enable mapreduce strict mode by setting below property to strict.

```
  <property>
  <name>hive.mapred.mode</name>
  <value>nonstrict</value>
  <description>
      The mode in which the Hive operations are being performed.
      In strict mode, some risky queries are not allowed to run. They include:
        Cartesian Product.
        No partition being picked up for a query.
        Comparing bigints and strings.
        Comparing bigints and doubles.
        Orderby without limit.
   </description>
```

8. Single Reduce for Multi Group BY

By enabling single reducer task for multi group by operations, we can combine multiple GROUP BY operations in a query into a single MapReduce job.

```
<property>
    <name>hive.multigroupby.singlereducer</name>
    <value>true</value>
    <description>
      Whether to optimize multi group by query to generate single M/R  job plan. If the mu
      common group by keys, it will be optimized to generate single M/R job.
    </description>
  </property>
```

9. Controls Parallel Reduce Tasks

We can control the number of parallel reduce tasks that can be run for a given hive query with below properties.

```
  <property>
    <name>hive.exec.reducers.bytes.per.reducer</name>
    <value>256000000</value>
    <description>size per reducer.The default is 256Mb, i.e if the input size is 1G, it wi
  </property>
  <property>
    <name>hive.exec.reducers.max</name>
    <value>1009</value>
    <description>
      max number of reducers will be used. If the one specified in the configuration parame
      negative, Hive will use this one as the max number of reducers when automatically de
    </description>
  </property>
```

we can also set the parallel reduce tasks to a fixed value with below property.

```
hive> set mapred.reduce.tasks=32;
```

orization

Vectorization feature is introduced into hive for the first time in hive-0.13.1 release only. By vectorized query execution, we can improve performance of operations like scans, aggregations, filters and joins, by performing them in batches of 1024 rows at once instead of single row each time.

We can enable vectorized query execution by setting below three properties in either hive shell or hive-site.xml file.

```
hive> set hive.vectorized.execution.enabled = true;
hive> set hive.vectorized.execution.reduce.enabled = true;
hive> set hive.vectorized.execution.reduce.groupby.enabled = true;
```

11. Enable Cost Based Optimization

Recent Hive releases provided the feature of cost based optimization, one can achieve further optimizations based on query cost, resulting in potentially different decisions: how to order joins, which type of join to perform, degree of parallelism and others.

cost based optimization can be enabled by setting below properties in hive-site.xml file.

And we can gather basic statistics about all columns in an employee table with below command in hive shell.

```
[illegible] employee COMPUTE STATISTICS FOR COLUMNS;
hive> ANALYZE TABLE employee COMPUTE STATISTICS FOR COLUMNS id, dept;
```

12. Use ORC File Format

Using ORC (Optimized Record Columnar) file format we can improve the performance of Hive Queries very effectively. Below picture on file format best depicts the power of ORC file file over other formats.

Create Tables with ORC File Format

We can create new hive table with ORC file format with just by adding **STORED AS ORC** clause to **CREATE TABLE** command in hive. Optionally we can provide compression techniques in **TBLPROPERTIES** clause.

es to ORC

Create a table with the same schema as the source table and **STORED AS ORC**, then we can submit below command to copy data from regular old table new ORC formatted table.

```
hive> INSERT OVERWRITE TABLE orc_emp SELECT * FROM emp;
```

Key	**Default**	**Notes**
`orc.compress`	`ZLIB`	Compression to use in addition to columnar compression (one of NONE, ZLIB, SNAPPY)
`orc.compress.size`	`262,144 (= 256 KiB)`	Number of bytes in each compression chunk
orc.stripe.size	`268,435,456 (= 256 MiB)`	Number of bytes in each stripe
`orc.row.index.stride`	`10,000`	Number of rows between index entries (must be >= 1,000)
`orc.create.index`	`true`	Whether to create inline indexes

Example ORC table creation:

```
hive> CREATE TABLE EMP_ORC (id int, name string, age int, address string)
    >  STORED AS ORC tblproperties ("orc.compress" = "SNAPPY");
hive> INSERT OVERWRITE TABLE EMP_ORC SELECT * FROM EMP;
```

Thus by using these 12 techniques we can improve the performance of Hive Queries.

Share this:

Share 7   Tweet

About Siva

Senior Hadoop developer with 4 years of experience in designing and architecture solutions for the Big Data domain and has been involved with several complex engagements. Technical strengths include Hadoop, YARN, Mapreduce, Hive, Sqoop, Flume, Pig, HBase, Phoenix, Oozie, Falcon, Kafka, Storm, Spark, MySQL and Java.

View all posts by Siva →

Leave a comment

Your email address will not be published. Required fields are marked *

Visual Text

File Edit View Insert Format Tools Table

Name *

Email *

Website

Post Comment

7 thoughts on “Hive Performance Tuning”

varun vikram

July 18, 2016 at 10:40 pm

Reply ↓

Can we actually run parallel reduce tasks in hadoop ?

Siva Post author

July 19, 2016 at 11:49 am

Reply ↓

Yes we can

Ann

August 31, 2016 at 3:30 pm

Reply ↓

Hi,

I have .hql file with a set of simple *select* queries in it. The .hql file size is 7,185 bytes (no.of queries = 20) and its taking nearly 1 hour for completion.

Sample query: select count distinct <field> from <table>

In few queries I also use join ,group by , order by etc

I would like to know if I can reduce the file execution time.

I tried implementing the below properties,

```
set hive.auto.convert.join=true;
set hive.auto.convert.sortmerge.join=true;
set hive.auto.convert.sortmerge.join.noconditionaltask=false;
set mapred.compress.map.output=true;
set mapred.output.compress=true;
set hive.exec.parallel=true;
set hive.vectorized.execution.enabled = true;
set hive.vectorized.execution.reduce.enabled = true;
```

But still I can't find any big change in execution time. Could you please help?

Mukesh

September 3, 2016 at 10:53 pm

Reply ↓

Hi Ann,

For : SELECT countDISTINCT FROM tab;
Try changing this to
SELECT count(1) FROM (SELECT DISTINCT field FROM tab) table;

This helps increasing underlying reducers instead of 1 and gain some through put.

Check if cardinal columns are there change their positions in GROUP BY clause if any..

Priyanka

Reply ↓

November 23, 2016 at 11:14 am

I need to optimize an insert overwrite statement in hive. Any help on how to do that?

Juan Pablo Rojas

Reply ↓

June 20, 2017 at 8:15 pm

The database is in SQL Server, the client uses ODBC connector (HortonWorks Hive 2.1) sends INSERT statements sequentially to the HiveServer2 server (I note that every time a statement is sent it opens and closes the connection), this generates a Really bad performance about 20 rows in 30 seconds. Optimize as much as possible the configuration of the HiveServer, hive-site.xml, map-redsite.xml and increase the memory of hiveserver2, however the performance does not improve.
It is true that Hive is Batch-oriented, however I find that this can be improved, I am very grateful for the help with this problem

Additionally any other architecture or tool to improve this process will be highly appreciated

Riya

Reply ↓

July 15, 2020 at 11:39 am

Hi,

I am trying to execute an insert overwrite statement in spark for a partitioned table. It is taking hours to execute.Can you please help me with this?

Search

Search

Core Hadoop

Big Data

Hadoop

Map Reduce

Hadoop Online Tutorial
A learning center for hadoop Eco System

Ecosystem Tools

Hive

Pig

HBase

Impala

Contact Me

please reach out to us on siv535@gmail.com or +91-9704231873

Recent Comments

- raviteja on Formula to Calculate HDFS nodes storage
- Durgesh Majeti on HDFS Web UI
- Brijesh Yadav on Run Example MapReduce Program
- ravikiran on Enable Compression in Hive
- Riya on Hive Performance Tuning

Hadooptutorial.info

Contat Us

Call Us On : +91-9704231873
Mail Us On : siv535@gmail.com

Email ID

Your email address

Send

Let's get Social :