

Best Practices

# Impala Performance Tuning Guidelines

Date published: 2019-12-10

Date modified:

The Cloudera logo is displayed in a bold, orange, sans-serif font. The word "CLOUDERA" is written in all caps, with a stylized 'E' that has three horizontal bars.

<https://docs.cloudera.com/>

# Legal Notice

© Cloudera Inc. 2023. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

# Contents

|  |           |
|--|-----------|
| <b>Overview.....</b>                                   | <b>5</b>  |
| <b>Kudu RPC.....</b>                                   | <b>5</b>  |
| <b>Setting up dedicated coordinators.....</b>          | <b>6</b>  |
| Load balancing for coordinators.....                   | 8         |
| <b>On-demand metadata and metadata management.....</b> | <b>8</b>  |
| Enabling on-demand metadata fetch.....                 | 8         |
| Enabling release of stale metadata.....                | 9         |
| Avoiding small files.....                              | 10        |
| Automatic metadata management.....                     | 10        |
| Coding in Spark for automatic metadata management..... | 12        |
| Manual metadata management.....                        | 12        |
| <b>Admission control.....</b>                          | <b>12</b> |
| Estimating memory limits.....                          | 15        |
| <b>Resource pool design.....</b>                       | <b>16</b> |
| <b>Table and column statistics.....</b>                | <b>18</b> |
| Setting statistics manually.....                       | 18        |
| <b>Tuning SQL queries.....</b>                         | <b>19</b> |
| <b>Recommended SET options for Impala.....</b>         | <b>19</b> |
| <b>Recommended configurations.....</b>                 | <b>20</b> |
| <b>Using Impala with Hue.....</b>                      | <b>20</b> |
| <b>Using Impala with BI tools.....</b>                 | <b>21</b> |
| <b>Appropriate file formats.....</b>                   | <b>21</b> |

|   |           |
|---|-----------|
| <b>Partitioning granularity recommendations.....</b>                    | <b>21</b> |
| <b>Addressing hotspotting.....</b>                                      | <b>21</b> |
| Detecting block skews.....  | 22        |
| <b>Minimizing overhead when transmitting results to clients.....</b>    | <b>23</b> |
| <b>Join query performance tuning.....</b>                               | <b>23</b> |
| <b>Query profiles.....</b>  | <b>24</b> |
| Execution summary.....  | 25        |
| Query timeline.....   | 25        |
| <b>Common scenarios for debugging queries using query profiles.....</b> | <b>26</b> |
| Memory limit exceeded.....  | 26        |
| Query runs slowly.....  | 26        |
| Admission control.....  | 27        |
| Client fetch wait timer.....  | 27        |
| Wrong join order.....   | 28        |
| Time and data skews.....  | 28        |

## Overview

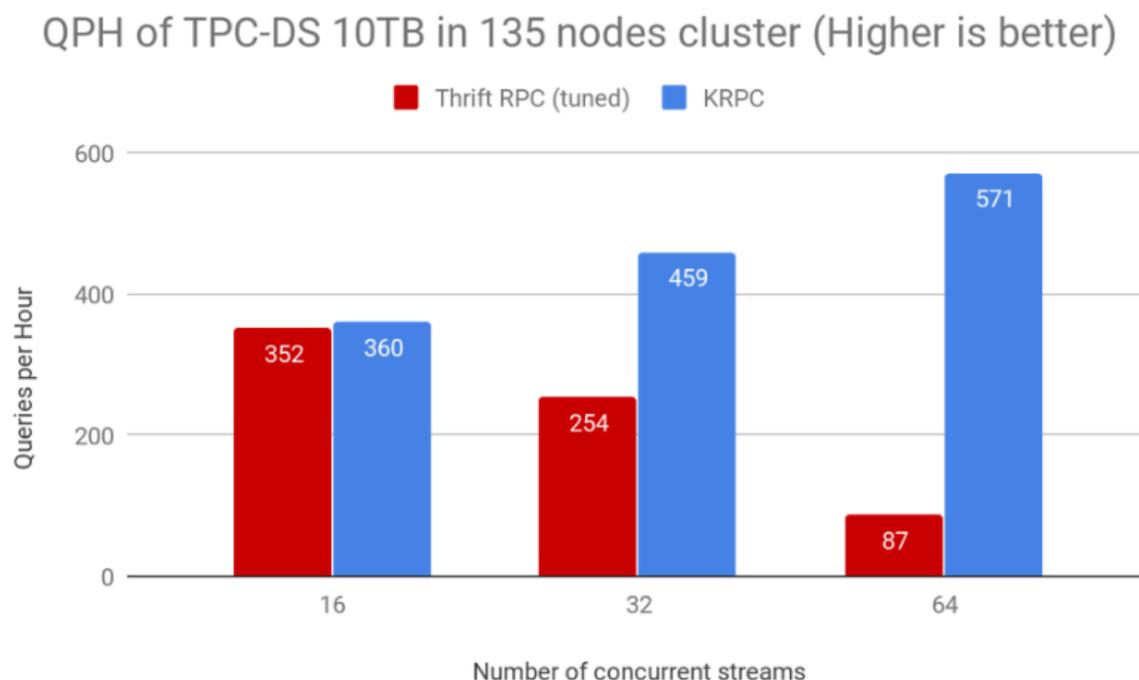
This document explains the factors that affect Impala query performance. In addition, it provides procedures for tuning, monitoring, and benchmarking Impala queries and other SQL operations to maximize Impala scalability. Scalability and performance go hand in hand. Improving query performance by reducing the disk I/O or memory used improves scalability by optimally using resources. When resources are used more efficiently, more queries can run in parallel.

## Kudu RPC

CDH 5.15, CDH 6.1, and later versions of Impala use a new network protocol called Kudu RPC (KRPC) for communication between the Impala brokers.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

KRPC replaces the older Thrift RPC protocol and significantly improves query performance. Cloudera recommends upgrading to CDH 5.15 and CDH 6.1 or newer versions to benefit from KRPC. With KRPC, queries can run 2-3 times faster and their success rate is higher:



KRPC main benefits:

- Reduces the total number of connections in a cluster.
- Reduces stress on the MIT KDC or on the Active Directory KDC.
- Supports connection multiplexing using one connection per direction between every pair of hosts

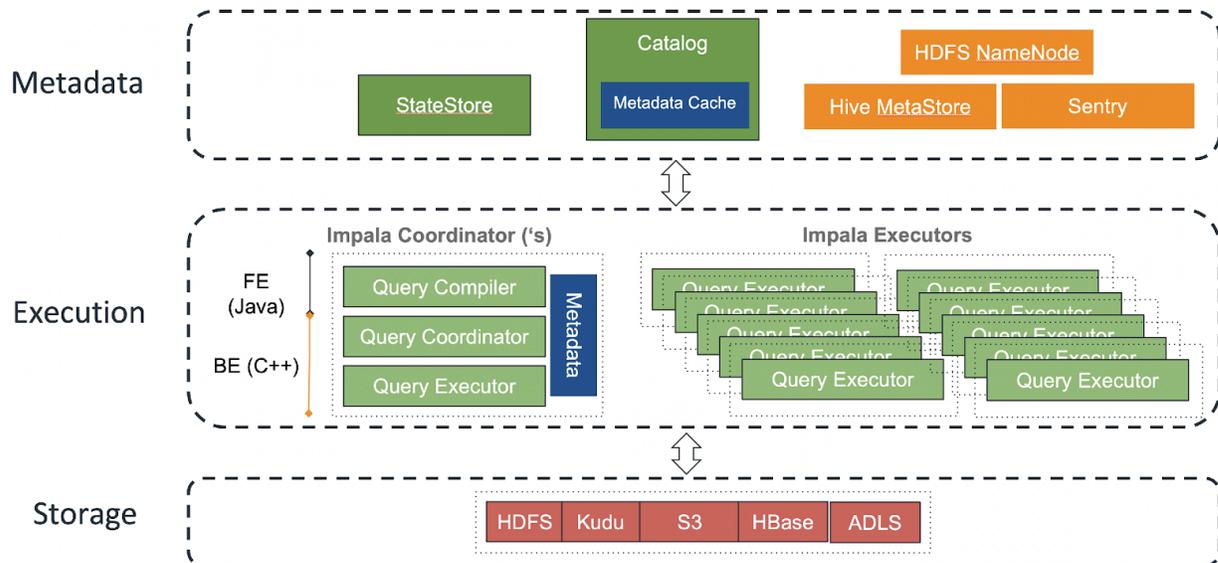
See the [Cloudera blog](#) for further details about KRPC.

## Setting up dedicated coordinators

Setting up dedicated coordinators helps improve performance on large clusters or for large Impala workloads.

By: [Manish Maheshwari](#), Data Architect and Data Scientist at Cloudera, Inc.

Impala can build execution plans and execute queries rapidly because it caches block and file metadata for HDFS tables on the Catalog Server and the query coordinator. When a new table is queried, the coordinator node requests the metadata from the catalog service daemon (catalogd), which in turn talks to the Hive Metastore (HMS) and the Namenode to retrieve the information:

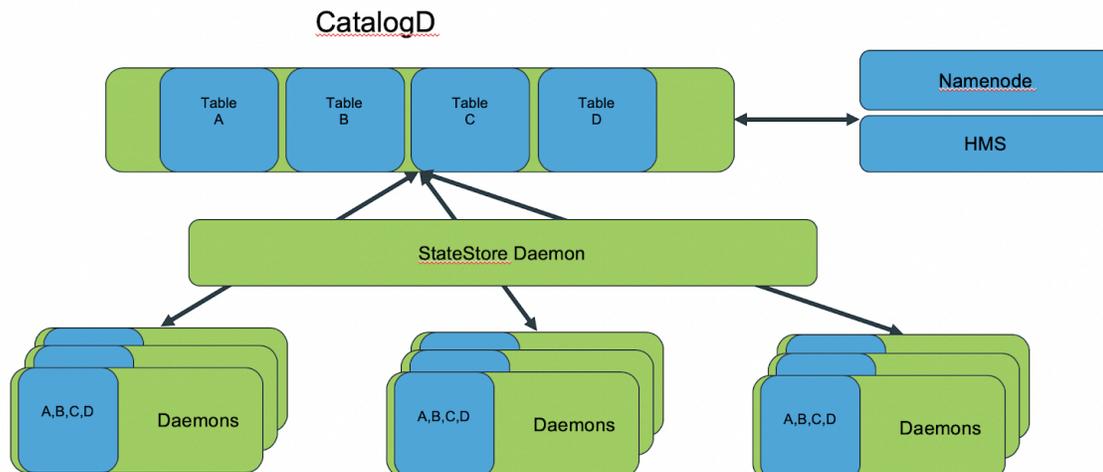


### Problem with typical coordinator setup:

On large clusters or for large Impala workloads, the typical coordinator setup can become problematic for the following reasons:

- Each coordinator caches metadata for all table partitions and data files. This requires coordinators to be configured with a large JVM heap size.
- The extra work required for the coordinator interferes with its capacity to perform query execution work due to network and CPU overhead for queries containing a large number of query fragments.

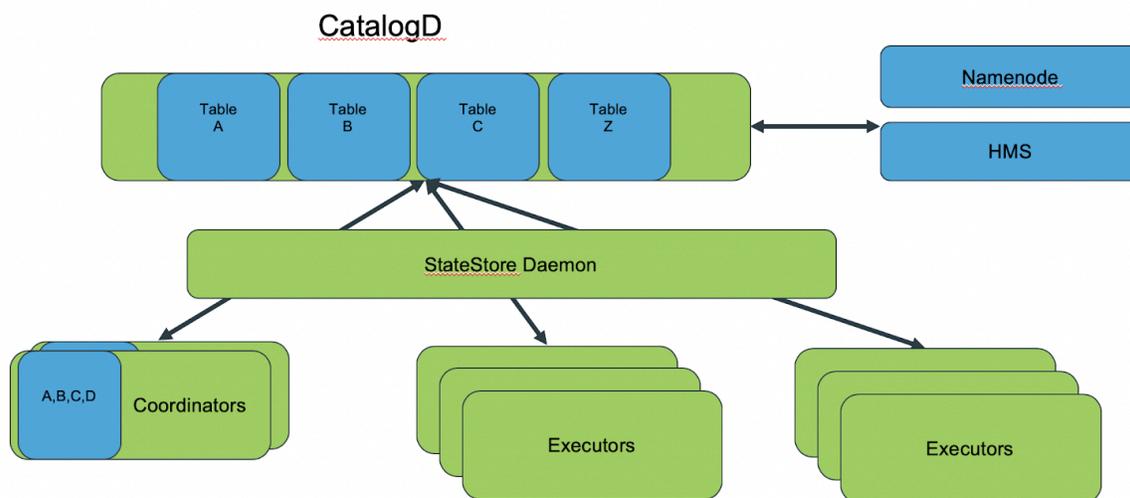
- Having a large number of hosts acting as coordinators can cause unnecessary network overhead because each of those hosts receives metadata from the Statestore daemon (statestored) for metadata updates>:



- Depending on the size of the Hive Metastore and the partitions, files, and blocks of the tables being queried, the in-memory catalog can become very large. This can cause pressure on the Impala daemons' memory.
- The soft limits imposed by the admission control feature are more likely to be exceeded when there is a large number of heavily loaded hosts acting as coordinators, which causes admission control limits to be exceeded.

#### Dedicated coordinator setup solution:

To solve this coordinator setup problem, separate roles for coordinators and executors as shown below:



#### How the dedicated coordinator setup works

- Run coordinator roles on edge nodes and gateway nodes. Run executors on the data nodes.
- Only coordinator nodes cache the metadata. They perform the task of query coordination and a few other tasks like final sorts for the queries. The executor roles read from HDFS, join tables, and some related tasks.
- With this configuration, the executor daemons cannot be connected through Impala shell or JDBC connections.

- Configure coordinator daemons with large enough JVM heap size to hold the metadata in memory. Configure For Impala daemons that are assigned a dedicated role as an executor only, configure those with the default JVM heap size. See [Impala Properties](#) for the suggested default JVM heap size to set for Impala daemons.

### Benefits of setting up dedicated coordinators

- Reduces memory usage on executor nodes.
- Avoids coordinator CPU bottlenecks caused by query execution tasks.
- Reduces the likelihood that admission control limits are exceeded.
- Reduces network utilization on the Statestore daemon by limiting metadata broadcast to only the coordinator nodes.
- Improves reliability and performance by reducing workload stress on coordinators.

## Load balancing for coordinators

Cloudera recommends setting up a load balancer for incoming connections to Impala.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

Load balancers such as [HaProxy](#) or [F5](#) usually work well to load balance incoming JDBC and Impala shell connections. The load balancer hostname should also be configured for Hue to connect to Impala.

The rule of thumb for load balancer configurations are as follows:

- Always configure your load balancer to use *balance source*, which uses the source IP address, to ensure that connections from one client are always routed to the same coordinator node.
- The load balancer timeout should be set to a value that is greater than the Impala idle session timeout. For example, if you are using HaProxy and set it up with a timeout of 5 minutes using `timeout server 300s`, ensure that the Impala idle timeout is set to less than 300 seconds.

## On-demand metadata and metadata management

The on-demand metadata feature streamlines metadata handling on the coordinators.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

Before the on-demand metadata feature was released, every coordinator kept a replica of all the metadata locally. This consumed a large amount of memory on each coordinator with no option to evict it. Metadata was sent by the Statestore to all the coordinators despite how the metadata was eventually used.

The on-demand metadata feature includes the following improvements:

- Coordinators pull metadata from the Catalog daemon (`catalogd`) for the required tables when they need it for query planning and cache it locally. The Statestore no longer transmits all the metadata for all the tables to each coordinator.
- Cached metadata gets evicted automatically under memory pressure treating the cache like an LRU (least recently used) buffer.
- Granularity of on-demand metadata fetch is at the partition level, so common use cases like add/drop partitions do not trigger unnecessary metadata transfers.

For more information about on-demand metadata management in Impala, see [Managing Metadata in Impala](#).

## Enabling on-demand metadata fetch

Use Cloudera Manager to enable on-demand metadata fetch for Impala.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

### To enable on-demand metadata fetch in Cloudera Manager:

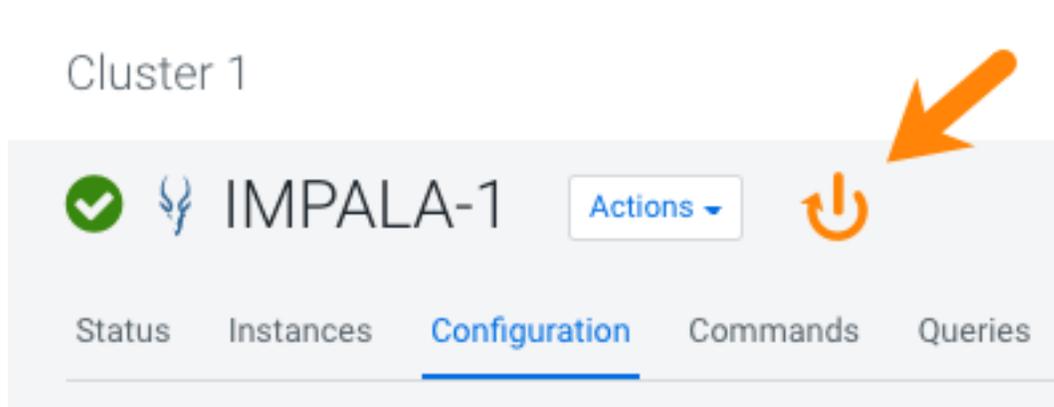
1. In Cloudera Manager Admin Console, click IMPALA.
2. Click the Configuration tab.
3. Search for catalogd.
4. Scroll down to locate the Catalog Server Command Line Argument Advanced Configuration Snippet (Safety Valve), and enter the following configuration information into the text box:

```
--catalog_topic_mode=minimal
```

5. Search for impalad.
6. Scroll down to locate the Impala Daemon Command Line Argument Advanced Configuration Snippet (Safety Valve), and enter the following configuration information into the text box:

```
--use_local_catalog=true
```

7. Click Save Changes at the bottom right corner of the page.
8. Scroll up to the top of the page and click the refresh icon to restart the services so your configuration changes can take effect:



### Enabling release of stale metadata

Use Cloudera Manager to enable the release of stale metadata in Impala.

By: [Manish Maheshwari](#), Data Architect and Data Scientist at Cloudera, Inc.

1. In Cloudera Manager Admin Console, click IMPALA.
2. Click the Configuration tab.
3. Search for catalogd.
4. Scroll down to locate the Catalog Server Command Line Argument Advanced Configuration Snippet (Safety Valve), and enter the following configuration information into the text box:

```
--invalidate_tables_timeout_s=28800
```

```
--invalidate_tables_on_memory_pressure=true
```

The above sets the Catalog daemon to release metadata after it has not been accessed (not used) in 28,800 seconds or 8 hours.

5. Search for impalad.

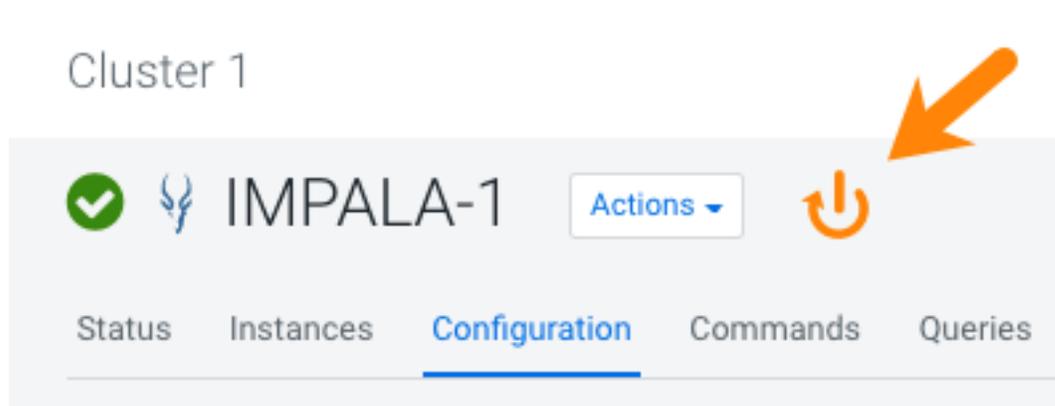
6. Scroll down to locate the Impala Daemon Command Line Argument Advanced Configuration Snippet (Safety Valve), and enter the following configuration information into the text box:

```
--invalidate_tables_timeout_s=28800
```

```
--invalidate_tables_on_memory_pressure=true
```

The above sets the Impala daemon to release metadata after it has not been accessed (not used) in 28,800 seconds or 8 hours.

7. Click Save Changes at the bottom right corner of the page.
8. Scroll up to the top of the page and click the refresh icon to restart the services so your configuration changes can take effect:



## Avoiding small files

To reduce the amount of memory used by the Catalog for metadata, avoid creating many small files in HDFS.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

Small files in HDFS can be caused by either having partitions that are too granular or by performing data ingestion too frequently. Cloudera recommends that you regularly compact small files. In Hive, you can compact small files with the following SQL commands:

```
SET hive.merge.mapfiles = true;
SET hive.merge.mapredfiles = true;
SET hive.merge.size.per.task = 256000000;
SET hive.merge.smallfiles.avgsize = 134217728;
SET hive.exec.compress.output = true;
SET parquet.compression = snappy;
INSERT OVERWRITE TABLE db_name.table_name SELECT * FROM db_name.table_name;
Run <Refresh Table> in impala after the Hive job finishes.
```

For tables with many partitions, change your partitioning strategy to partition in a less granular way. For example, partition by year/month instead of by year/month/day. If you are doing inserts with Impala, use `/* +SHUFFLE */` optimizer hints, which add an exchange node before writing the data. Using this hint, only one node writes to a partition at a time, reducing the number of files written. See [Optimizer Hints in Impala](#) for more information.

## Automatic metadata management

Impala automatic metadata management feature uses notifications to refresh data after changes.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

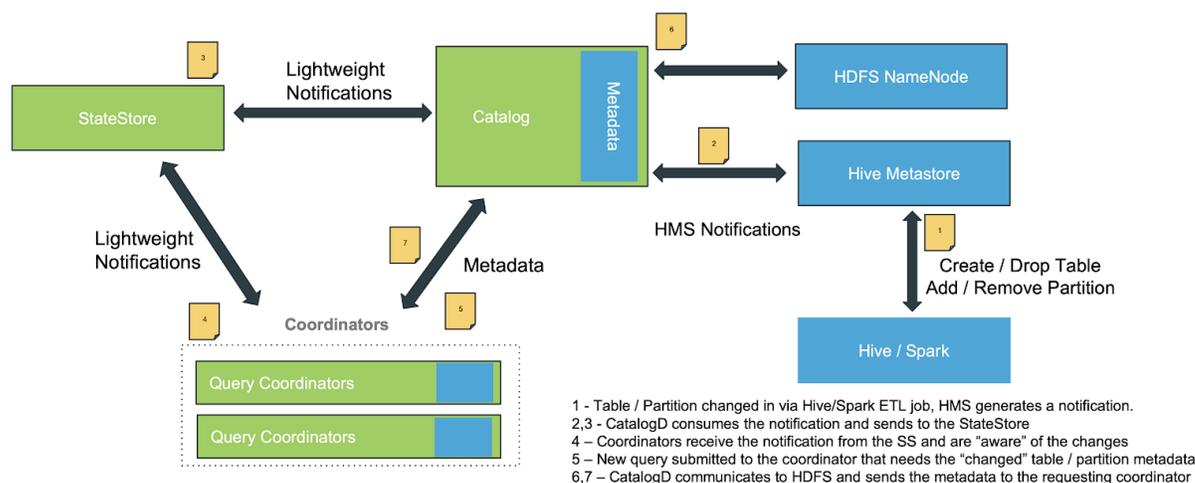
When tools such as Hive and Spark are used to process raw data ingested into Hive tables, new Hive metastore metadata (in the form of databases, tables, and partitions) and file system metadata (in the form of new files in existing partitions and tables) are generated. In previous versions of Impala, to pick up this new information, Impala users had to manually issue an `INVALIDATE` or `REFRESH` command. Now, there is a new feature called automatic metadata management. Automatic metadata management works by using Hive metastore notifications which performs the following:

- Invalidates tables when it receives an `ALTER TABLE` event.
- Refreshes the partition when it receives an `ALTER TABLE ADD | DROP PARTITION` event.
- Adds the tables or databases when it receives a `CREATE TABLE` or `CREATE DATABASE` event.
- Removes tables from the Catalog (catalogd) when it receives a `DROP TABLE` or `DROP DATABASE` events.
- Refreshes table and partitions when it receives `INSERT` events.

When there are database-level changes, the following types of changes are supported:

- Database properties
- Comments on the database
- Owner of the database
- Default location of the database

**Figure 1: Hive metastore notifications**



To control this feature, use the `--hms_event_polling_interval_s` flag on the `catalogd`. When set to a positive value, it enables Hive metastore event polling. The recommended value to set this flag to is under 5 seconds.

The automatic metadata management feature does not support files or partitions that have been manually added to HDFS by using Spark, DistCp, or HDFS Put because no Hive metastore notification is generated for these operations. To handle these scenarios, use one of the following methods:

- Use the `LOAD DATA` command to handle the metadata changes, or
- Run a `REFRESH [db_name.]table_name [PARTITION...]` command or an `ALTER TABLE table_name RECOVER PARTITIONS` command.

If you need to disable automatic metadata management for certain tables or databases, set the following properties:

```
CREATE DATABASE <db_name> DBPROPERTIES ('impala.disableHmsSync'='true');
CREATE TABLE <tab_name> WITHTBLPROPERTIES ('impala.disableHmsSync'='true' |
'false');
```

## Coding in Spark for automatic metadata management

The Spark API, which saves data to a specified location, does not generate events in the Hive metastore so it is not supported by automatic metadata management.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

For example here is an example of a Spark Scala API code, which does not generate events in the Hive metastore:

```
Seq((1, 2)).toDF("i",  
"j").write.save("/user/hive/warehouse/spark_etl.db/customers/date=01012019")
```

Instead, use the below Spark SQL code to ensure that Hive metastore events are generated and sent to the metastore:

```
Spark.sql(" INSERT OVERWRITE TABLE xxx PARTITION (date = , ...) as select * f  
rom spark_dataframe" )
```

## Manual metadata management

If you are not using CDH 6.2 or 5.16, you must manage metadata manually for ETL operations that run in Hive on Spark or third party ETL tools.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

For manual metadata management, Cloudera recommends that metadata operations be handled as part of ETL code. This ensures that BI end users need not refresh tables when new data is added. After data is loaded with Hive on Spark, use either the Impala shell or the Impala JDBC connector in Spark to connect to Impala and perform the following operations:

- `REFRESH <table_name>` or `REFRESH <table_name> PARTITION`
  - Reloads metadata for the table and incrementally reloads file and block metadata from the HDFS NameNode.
  - Should be run when adding, removing, or overwriting files into existing partitions by using Hive on Spark.
- `ALTER TABLE <table_name> RECOVER PARTITIONS`
  - Scans HDFS to check if any new partition directories were added and cache block metadata for those files.
  - Should be run when new partitions are added to existing tables. Note that it is faster than using `REFRESH <table_name>`.
- `INVALIDATE METADATA <table_name>`
  - Runs asynchronously to discard the loaded metadata catalog cache for the table. Metadata loading for tables is triggered by any subsequent queries.
  - Run `INVALIDATE METADATA` when new tables are created or dropped by external tools such as Hive on Spark or after running the HDFS balancer.

See the [Impala SQL Reference](#) for details on these SQL statements.

## Admission control

The admission control feature controls the number of concurrent queries running on the cluster, which avoids out-of-memory issues on busy clusters.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

Once the limit that you set in admission control has been reached, subsequent queries are queued and wait for running queries to finish before they run. You can configure the amount of time that queries can wait. Using admission control, you can set the default memory that can be used by each query in a pool. If you do not set a default memory

limit, Impala uses heuristics based on table and column statistics to determine the amount of memory that each query requires.

Cloudera recommends that you set the number of concurrently running queries between 40-60. This loosely corresponds to the number of threads on each of the Impala daemons.

Best practices for using admission control are:

- Ensure that admission control is enabled by checking the following setting in Cloudera Manager:

**Enable Impala Admission Control**

IMPALA-1 (Service-Wide)

---

- Ensure that the Minimum Query Memory Limit and the Maximum Query Memory Limit, or the Default Query Memory Limit is set for all pools:

Impala Admission Control Configuration

Resource Pools   Scheduling Rules   Placement Rules

Each pool can support different limits, and can be configured to allow only a certain set of users and groups to access the pool.

3 running Impala Daemons are configured with a total of 603.7 GiB of memory.

Create Resource Pool   Default Settings

| Name               | Max Memory | Max Running Queries | Max Queued Queries | Queue Timeout | Minimum Query Memory Limit | Maximum Query Memory Limit | Clamp MEM_LIMIT Query Option | Default Query Memory Limit |
|--------------------|------------|---------------------|--------------------|---------------|----------------------------|----------------------------|------------------------------|----------------------------|
| root.large_queries | 200 GiB    | 5                   | 10                 | 10 minute(s)  | 3 GiB                      | 5 GiB                      | true                         | No Default                 |
| root.small_queries | 300 GiB    | 10                  | 20                 | 10 minute(s)  |                            |                            |                              | 5 GiB                      |

You can check these settings in Cloudera Manager. From the Cloudera Manager home page, click Clusters in the left navigation tree and select Impala Admission Control Configuration:

The screenshot shows the Cloudera Manager interface. On the left is a dark navigation sidebar with the Cloudera logo and a search bar. Below the search bar are menu items: Clusters, Hosts, Diagnostics, Audits, Charts, Replication, and Administration. The 'Clusters' menu is expanded, showing a list of services: Data Analytics Studio, HBASE-1, HDFS-1, HIVE-1, HIVE\_ON\_TEZ-1, HUE-1, IMPALA-1, KAFKA-1, KUDU-1, LIVY-1, OOZIE-1, OZONE-1, and SOLR-1. An orange arrow points to the 'IMPALA-1' service. On the right side of the interface, under the 'Cluster 1' header, there is a list of configuration options: Hosts, Roles, Host Templates, Parcels, Send Diagnostic Data, Reports, Utilization Report, IMPALA-1 Queries, YARN Applications, Impala Admission Control Configuration (highlighted with an orange arrow), and Static Service Pools.

In CDH 6.2, admission control settings have also been added to the Impala Web UI. See [Troubleshooting with the Impala Web UI](#) in the product documentation for details.

- If the default memory limit is set, query profiles show the following message:

```
Query Options (set by configuration): MEM_LIMIT=2147483648,REQUEST_POOL=RPIBR
Query Options (set by configuration and planner): MEM_LIMIT=2147483648,REQUEST_POOL=RPIBR,MT_DOP=0
```

- If the default memory limit is set too low and the query times out and you receive a “Memory limit exceeded” error message, rerun the query using the SET MEM\_LIMIT option. For example:

```
SET MEM_LIMIT = 3gb;
```

For details about this SET statement, see [MEM\\_LIMIT Query Option](#).

- The total maximum memory set across all of the pools should be equal to the total memory that is allocated to the Impala service.

## Estimating memory limits

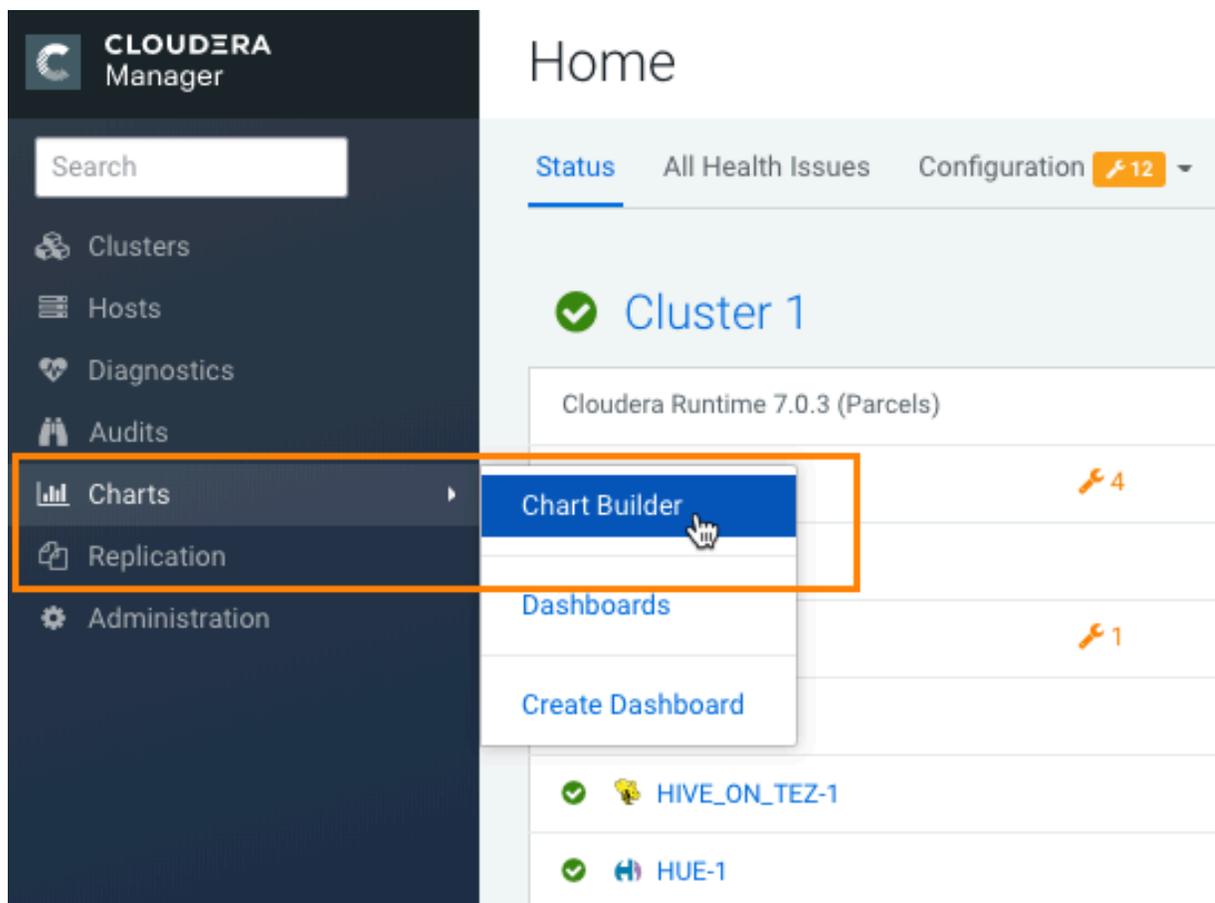
Use Cloudera Manager Chart Builder to determine memory limits for resource pools.

By: [Manish Maheshwari](#), Data Architect and Data Scientist at Cloudera, Inc.

To identify what is the appropriate minimum, maximum, and default memory limit for each of the pools, Cloudera recommends that you run real queries on your datasets, and then review the memory usage during peak periods of this test.

After you have run a set of your real queries on your datasets, use Cloudera Manager Chart Builder to view charts that reflect memory usage:

1. On the Cloudera Manager Admin Console home page, from the left navigation tree, select **Charts** > **Chart Builder** :



2. On the Chart Builder page, enter the following query, and then click Build Chart:

```
select memory_per_node_peak from IMPALA_QUERIES where
```

```
service_name="impala"
```

Chart Builder then returns a chart that might look similar to the following example:

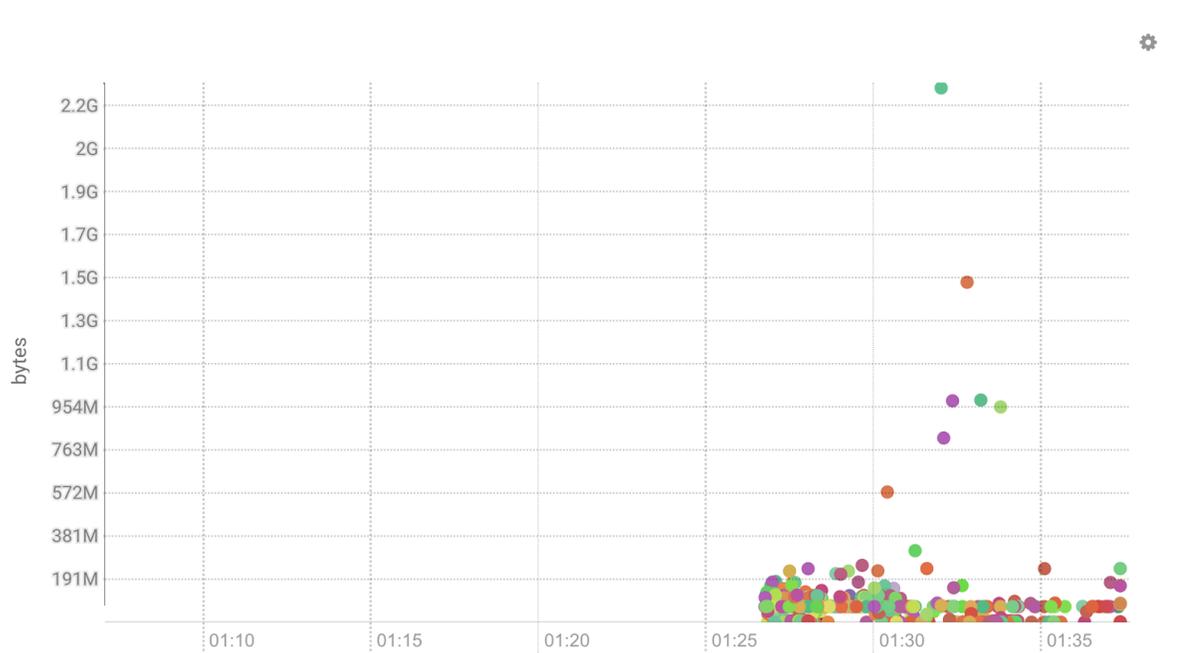
### Per Node Peak Memory Usage

Query

```
select memory_per_node_peak from IMPALA_QUERIES where service_name = "impala"
```

Data Granularity

Auto



This type of chart can be used to determine the memory limits for a particular pool. Note that most queries use less than 1.2 GB per node. See [Resource Management](#) for more information about Impala admission control.

## Resource pool design

Allocating memory across resource pools is an important performance tuning task.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

With admission control enabled, the most common approach used to allocate the total amount of Impala memory across all of the pools is to create a pool for each tenant as shown in the following diagrams:

|                 | Executor 1 | Executor 2 | ... | ... | ... | Executor 10 |
|-----------------|------------|------------|-----|-----|-----|-------------|
| <b>Tenant 1</b> |            |            |     |     |     |             |
| <b>Tenant 2</b> |            |            |     |     |     |             |
| <b>Tenant 3</b> |            |            |     |     |     |             |
| <b>Tenant 4</b> |            |            |     |     |     |             |
| <b>Tenant 5</b> |            |            |     |     |     |             |
| <b>Tenant 6</b> |            |            |     |     |     |             |
| <b>Tenant 7</b> |            |            |     |     |     |             |
| <b>Tenant 8</b> |            |            |     |     |     |             |

However this strategy is usually counterproductive because of the following reasons:

- Unused memory in one tenant cannot be used by other tenants.
- Busy tenants queue up queries in admission control, causes an overall slowdown in query execution.
- Small tenants that run large queries spill to disk until the spill to disk limit is reached and the memory limit is reached.

Instead, Cloudera recommends that you design resource pools by query sizes as shown in the following diagram:

|                       | Executor 1 | Executor 2 | ... | ... | ... | Executor 10 |
|-----------------------|------------|------------|-----|-----|-----|-------------|
| <b>Small Queries</b>  |            |            |     |     |     |             |
| <b>Large Queries</b>  |            |            |     |     |     |             |
| <b>Medium Queries</b> |            |            |     |     |     |             |

This method ensures that all memory allocated to Impala is optimally used. Also, by using this method, all users and groups have access to all the pools and users can choose the appropriate pool according to memory sizes.

## Table and column statistics

Table and column statistics help Impala generate optimal query plans using table sizes and the degree of cardinality in columns.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

The statistics information is stored in the Hive metastore database. You run the COMPUTE STATS statement to collect and set table-level and partition-level row counts as well as column statistics for a particular table. However, running this statement is very CPU-intensive. Based on the number of rows, number of data files, the total size of the data files, and the file format that are involved, Cloudera recommends the following strategies for computing statistics:

- Missing stats can be identified by using the following SHOW commands:

```
SHOW TABLE STATS [database_name.]table_name
SHOW COLUMN STATS [database_name.]table_name
```

For more information about running SHOW commands, see [SHOW Statement](#).

- Table and column statistics are also recorded in the query profile as shown below:

```
F00:PLAN FRAGMENT [RANDOM] hosts=5 instances=5
00:SCAN HDFS [telco_db.call_trace_hist_bucketed, RANDOM]
  partitions=284/3396 files=916 size=206.74GB
  predicates: (calling_number = '734214' OR called_number = '734214')
  table stats: unavailable
  columns missing stats: date_time, msisdn, calling_number, called_number, duration, calling_location, called_location
  mem-estimate=616.00MB mem-reservation=0B
  tuple-ids=0 row-size=99B cardinality=391672994
```

- Do not compute statistics for tables that are not used in joins or for Impala queries. For example, statistics on a staging table that feeds into a data warehouse layer table one-to-one with no joins does not require statistics.
- Limit running COMPUTE STATS only to columns that are involved in filters, join conditions, or GROUP BY clauses. For example, running COMPUTE STATS in the following situation makes sense:

```
COMPUTE STATS wide_table [ join_column_a, join_column_b ]
```

- Rerun COMPUTE STATS only when there is over 30% data change in the data.
- Cloudera recommends that you run COMPUTE STATS in off-peak hours, on weekends, or at night.
- When you reload a complete table or partition, where the number of rows and distinct values for each column is relatively unchanged from before, there is no need to rerun COMPUTE STATS.

Also see [Table and Column Statistics](#) for more information about table and column statistics.

## Setting statistics manually

For tables and partitions where the statistics do not change or are already known, statistics can be set manually.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

Manually set statistics as follows:

- Set the total number of rows in a table:

```
ALTER TABLE <table_name> SET TBLPROPERTIES('numRows'='new_value', 'STATS_GENERATED_VIA_STATS_TASK'='true');
```

- Set the total number of rows for a specific partition:

```
ALTER TABLE <table_name> PARTITION (keycol1=value_1,keycol2=value_2...)
```

```
SET TBLPROPERTIES('numRows'='new_value', 'STATS_GENERATED_VIA_STATS_TASK'='true');
```

- Set column statistics:

```
ALTER TABLE <table_name> SET COLUMN STATS <col_name> ('numDVs'='100');
```

Compute the numDVs values by running the following query:

```
SELECT NDV(col_name) FROM <table_name>;
```

## Tuning SQL queries

While SQL query tuning is a broad subject overall, Cloudera recommends the following SQL best practices.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

- Hadoop and Impala are best suited for star schema data models over third normal form (3NF) models. While Impala can work efficiently with 3NF models, the lesser number of joins and wider tables used in star schema models typically corresponds to faster query execution times.
- Use integer join keys rather than character or data join keys.
- Always use columns of the varchar data type rather than of the character data type.
- Avoid using either implicit or explicit casts.

## Recommended SET options for Impala

Following are the recommended SET options for the best performance with Impala.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

- Always use the SET MEM\_LIMIT query option. See [MEM\\_LIMIT Query Option](#) for details.
- Use the APPX\_COUNT\_DISTINCT query option to improve performance of COUNT(DISTINCT) queries by converting them to use NDV() function calls over linear counting. Note that APPX\_COUNT\_DISTINCT is approximately 98% accurate for smaller data sets and over 99% accurate for larger datasets. See [APPX\\_COUNT\\_DISTINCT Query Option](#) for details.
- Always use a compression algorithm that both encodes and decodes (codec). Cloudera recommends that you use LZ4 or ZSTD with Impala version 3.3 and onwards. Use snappy for earlier versions of Impala.
- Do not use the DISABLE\_CODEGEN query option because it can significantly slow query execution. Instead, use the DISABLE\_CODEGEN\_ROWS\_THRESHOLD query option, which automatically disables codegen for small queries. For details, see [DISABLE\\_CODEGEN Query Option](#).
- Set the EXEC\_SINGLE\_NODE\_ROWS\_THRESHOLD query option to 1000 rows for faster execution of simple queries or queries that contain LIMIT clauses. See [EXEC\\_SINGLE\\_NODE\\_ROWS\\_THRESHOLD Query Option](#) for details.
- Set the NUM\_ROWS\_PRODUCED\_LIMIT query option for interactive user pools or in Hue to limit the rows produced by interactive queries. See [NUM\\_ROWS\\_PRODUCED\\_LIMIT Query Option](#) for details.
- Do not use the SYNC\_DDL = true query option. Instead, use your load balancer configurations to ensure that clients are always directed to the same query coordinators.
- Set the EXEC\_TIME\_LIMIT\_S query option to a value such as 1,200 seconds (20 minutes). This query option causes an executing queries to be canceled if it is still executing when the time limit expires. If a query is canceled, investigate the query plan to identify the reason for the slowness. See [EXEC\\_TIME\\_LIMIT\\_S Query Option](#) for details.

## Recommended configurations

Following are the recommended configuration setting for the best performance with Impala.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

- Set the `--use_local_tz_for_unix_timestamp_conversions` startup flag and the `--convert_legacy_hive_parquet_utc_timestamps` startup flag both to true. Setting these startup flags to true ensures that the timestamps between Hive and Impala match. See [TIMESTAMP Data Type](#) for more details.
- Always set the `--idle_session_timeout` and the `--idle_query_timeout` timeouts for the Impala daemon (`impalad`). Ensure that the setting for `idle_session_timeout` is less than the setting for the timeout set for your load balancer. See [Setting the Idle Query and Idle Session Timeouts for impalad](#) for details.
- Set the `--fe_service_threads` startup option for the Impala daemon (`impalad`) to 256. This option specifies the maximum number of concurrent client connections allowed. See [Startup Options for impalad Daemon](#) for details.
- Increase the `--num_metadata_loading_threads` startup option to 64 to improve metadata loading performance. See [Configuring Impala Startup Options through Cloudera Manager](#) for more information.

## Using Impala with Hue

Following are some recommended configurations that give you the best performance when you use Hue with Impala.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

- Always connect to the Impala load balancer, for example HaProxy, and not to a single coordinator. This helps to avoid the *hotspotting* issue on a single coordinator. *Hotspotting* occurs when too many tasks touch the same data, potentially overloading the node or coordinator.
- Set the `querycache_rows` configuration property to a lower value than its default setting of 50000 according to your requirements. This configuration property sets the number of initial rows of a result set that Impala caches in order to support re-fetching them.
- Set the `close_queries` configuration property to true. This property causes Hue to try to close an Impala query when the user leaves the editor page so it frees all Impala query resources. However, it makes query results for that user inaccessible.
- Set `query_timeout_s` and `session_timeout_s` configuration properties. You can use these properties to set timeouts for query execution and for sessions in Hue, which causes queries and sessions to be cancelled when the timeout period expires. They are set in seconds.

### To set these configuration properties in Cloudera Manager:

1. On the Cloudera Manager home page, select HueConfiguration .
2. Search for the Hue Service Advanced Configuration Snippet (Safety Valve) for `hue_safety_valve.ini`, and then append the following configuration information into the text box after any other configurations listed there:

```
[impala]
querycache_rows=<number_of_rows>
close_queries=true
query_timeout_s=<number_of_seconds>
session_timeout_s=<number_of_seconds>
```

3. Click Save Changes and restart the service.

## Using Impala with BI tools

Following are recommendations for using BI tools with Impala that give you the best performance.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

- Always enable LDAP authentication for BI tools to conveniently connect to Impala rather than using Kerberos authentication.
- Use caching on BI tools to serve results from the same query to multiple users.

## Appropriate file formats

Following are recommendations for which file formats provide the best performance in Impala.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

- For BI queries, the Parquet file format performs best because of its combination of columnar storage layout, compression, and encoding. The default setting for COMPRESSION\_CODEC is `snappy` compression, but `GZip` compression is also supported.
- Impala also supports reading ORC file formats from version 2.12 and onwards, however expect query performance with ORC tables to be slower than it is with Parquet tables.
- Text formats can be used when all columns are retrieved from a table. However, because compression on text is lower, HDFS I/O could be longer than when you use the Parquet file format.

## Partitioning granularity recommendations

Following are recommendations for table partitioning granularity that provides the best performance in Impala.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

- Choose a partitioning strategy that ensures there is at least 256 MB of data in each partition.
- Over-partitioning causes query planning to take longer than necessary because Impala prunes the unnecessary partitions, which results in small files in each partition.
- Cloudera recommends that you keep the number of partitions in tables under 30,000.
- Always use integer data types for partition key columns:
  - Partition key values are turned into HDFS directory names so you can minimize memory usage by using numeric values for common partition key fields such as YEAR, MONTH, and DAY.
  - Use the smallest integer data type that holds the appropriate range of values. Typically, TINYINT for MONTH and DAY, and SMALLINT for YEAR. Use the EXTRACT() function to pull out individual date and time fields from a TIMESTAMP value, and CAST() the return value to the appropriate integer data type.

## Addressing hotspotting

Following recommendations can address "hotspotting," which is what happens when many queries access the same node, causing it to become overloaded.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

When data for a particular query or the overall Impala workload is concentrated on a limited number of nodes, those nodes can "hotspot" due to scan fragments. For example, if you have a small dimension table that fits into a single HDFS block, multiple queries run the scan fragment on the same node causing it to get overloaded.

To avoid this hotspotting:

- For small dimension tables, increase the HDFS replication factor by using the HDFS File System shell command `hdfs dfs -setrep` options. See [setrep](#) in the command reference for more information.
- For fact table partitions that are heavily queried, the replication factor can be set temporarily. For example, you can set it for 7 days using the above `setrep` command. Then after 7 days have elapsed, you can reset the replication factor to 3.
- Impala planned behaviour can be changed by setting the following query options:
  - `SET SCHEDULE_RANDOM_REPLICA=true` See [SCHEDULE\\_RANDOM\\_REPLICA Query Option](#) for details.
  - `SET REPLICA_PREFERENCE=REMOTE` See [REPLICA\\_PREFERENCE Query Option](#) for details.
- HDFS caching can be used to cache block replicas. This causes the Impala scheduler to randomly pick a node that is hosting a cached block replica for the scan. See [Using HDFS Caching with Impala](#) for more information.

## Detecting block skews

Following are recommendations for detecting block skews, which are slow hosts that impact Impala performance.

By: [Manish Maheshwari](#), Data Architect and Data Scientist at Cloudera, Inc.

The most convenient way to detect a block skew or a “slow-host” issue is to compare the Avg Time to the Max Time in the execution summary section of the query plan:

| ExecSummary:<br>Operator | #Hosts | Avg Time  | Max Time  | #Rows   | Est. #Rows | Peak Mem  | Est. Peak Mem | Detail                          |
|--------------------------|--------|-----------|-----------|---------|------------|-----------|---------------|---------------------------------|
| 30: EXCHANGE             | 1      | 91.088us  | 91.088us  | 9       | 183.91K    | 0         | -1.00 B       | UNPARTITIONED                   |
| 15: AGGREGATE            | 104    | 1.142ms   | 1.699ms   | 9       | 183.91K    | 10.32 MB  | 10.00 MB      | FINALIZE                        |
| 29: AGGREGATE            | 104    | 1.134ms   | 1.430ms   | 9       | 183.91K    | 2.72 MB   | 10.00 MB      | FINALIZE                        |
| 28: EXCHANGE             | 104    | 66.080us  | 829.177us | 936     | 183.91K    | 0         | 0             | HASH(CASE WHEN t_6.timestamp... |
| 14: AGGREGATE            | 104    | 5.525ms   | 9.339ms   | 936     | 183.91K    | 5.02 MB   | 10.00 MB      | STREAMING                       |
| 13: HASH JOIN            | 104    | 180.884ms | 779.913ms | 2.08M   | 2.59M      | 2.75 MB   | 588.23 KB     | INNER JOIN, PARTITIONED         |
| --27: EXCHANGE           | 104    | 196.808us | 1.063ms   | 245.47K | 245.47K    | 0         | 0             | HASH(t_10.datadate,t_10.hour)   |
| 25: AGGREGATE            | 1      | 104.257ms | 104.257ms | 245.47K | 245.47K    | 146.59 MB | 14.94 MB      | FINALIZE                        |
| 24: EXCHANGE             | 1      | 6.728ms   | 6.728ms   | 245.47K | 245.47K    | 0         | 0             | HASH(t_10.datadate,t_10.hou...  |
| 08: AGGREGATE            | 1      | 96.275ms  | 96.275ms  | 245.47K | 245.47K    | 11.82 MB  | 14.94 MB      | STREAMING                       |
| 06: UNION                | 1      | 37.413ms  | 37.413ms  | 245.47K | 245.47K    | 395.58 KB | 0             | 0                               |
| 07: SCAN HDFS            | 1      | 1s060ms   | 1s060ms   | 245.47K | 245.47K    | 1.03 MB   | 448.00 MB     | bi_dw.dim_date_hour             |
| 26: EXCHANGE             | 104    | 4.722ms   | 8.463ms   | 2.08M   | 2.59M      | 0         | 0             | HASH(fact_booking_payments...   |
| 12: HASH JOIN            | 104    | 5s555ms   | 15s223ms  | 2.08M   | 2.59M      | 2.95 MB   | 8.76 MB       | INNER JOIN, BROADCAST           |
| --23: EXCHANGE           | 104    | 157.702us | 370.409us | 1.34K   | 245.47K    | 0         | 0             | BROADCAST                       |
| 22: AGGREGATE            | 1      | 67.308ms  | 67.308ms  | 1.34K   | 245.47K    | 18.57 MB  | 10.00 MB      | FINALIZE                        |
| 21: EXCHANGE             | 1      | 4.131ms   | 4.131ms   | 245.47K | 245.47K    | 0         | 0             | HASH(t_5.hour,t_5.timestamp...  |
| 05: AGGREGATE            | 1      | 59.959ms  | 59.959ms  | 245.47K | 245.47K    | 11.57 MB  | 10.00 MB      | STREAMING                       |
| 03: UNION                | 1      | 23.736ms  | 23.736ms  | 245.47K | 245.47K    | 271.02 KB | 0             | 0                               |
| 04: SCAN HDFS            | 1      | 1s761ms   | 1s761ms   | 245.47K | 245.47K    | 634.92 KB | 448.00 MB     | bi_dw.dim_date_hour             |
| 11: HASH JOIN            | 104    | 45.015ms  | 55.188ms  | 84.95M  | 2.59M      | 1.52 MB   | 38.00 B       | INNER JOIN, BROADCAST           |
| --20: EXCHANGE           | 104    | 12.637us  | 20.674us  | 1       | 1          | 0         | 0             | BROADCAST                       |
| 19: AGGREGATE            | 2      | 2.879ms   | 4.322ms   | 1       | 1          | 2.40 MB   | 10.00 MB      | FINALIZE                        |
| 18: EXCHANGE             | 2      | 5.471us   | 7.427us   | 1       | 1          | 0         | 0             | HASH(dim_web_store_t15.web...   |
| 10: AGGREGATE            | 2      | 10.070ms  | 18.604ms  | 1       | 1          | 1.50 MB   | 10.00 MB      | STREAMING                       |
| 09: SCAN HDFS            | 2      | 72.566ms  | 125.675ms | 1       | 1          | 238.00 KB | 32.00 MB      | bi_dw.dim_web_store dim_web...  |
| 02: HASH JOIN            | 104    | 527.151ms | 573.180ms | 86.23M  | 5.18M      | 153.64 MB | 11.06 MB      | INNER JOIN, PARTITIONED         |
| --17: EXCHANGE           | 104    | 33.998ms  | 55.734ms  | 70.37M  | 4.09M      | 0         | 0             | HASH(fact_booking.booking_i...  |
| 01: SCAN HDFS            | 104    | 1s653ms   | 4s287ms   | 70.37M  | 4.09M      | 34.62 MB  | 1.17 GB       | bi_dw.fact_booking fact_boo...  |
| 16: EXCHANGE             | 104    | 45.147ms  | 179.170ms | 104.36M | 120.51M    | 0         | 0             | HASH(fact_booking_payments...   |
| 00: SCAN HDFS            | 104    | 2s175ms   | 4s584ms   | 104.36M | 120.51M    | 74.10 MB  | 352.00 MB     | bi_dw.fact_booking_payments...  |

For each phase of the query, there is an Avg Time and a Max Time value, along with #Hosts, which indicates how many hosts are involved in that phase of the query. For all query phases involving more than one host, look for cases where the maximum time is substantially greater than the average time.

Slow hosts can be caused by a variety of reasons. To avoid slow hosts:

- Ensure that all Impala nodes have the same configuration.
- Ensure that other workloads, which are not run by Impala, are evenly distributed by setting the `yarn.scheduler.fair.assignmultiple` property to `false` in the `yarn-site.xml`.
- Ensure that HDFS data is balanced by running the HDFS balancer utility during the off-peak hours and then run the `INVALIDATE METADATA` on Impala metadata after you perform the rebalancing to refresh the metadata in the Hive metastore. See [HDFS Balancers](#) and [INVALIDATE METADATA Statement](#) for more information.

- Ensure that the HDFS block sizes are optimal. For example, use 256MB blocks and use [snappy](#) compression in a splittable block, which is preferable to [GZip](#).

## Minimizing overhead when transmitting results to clients

Always try to minimize the overhead incurred when transmitting query results back to clients.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

Use techniques such as:

- Always use the LIMIT clause in queries to sample data. See [LIMIT Clause](#) for more information.
- Use the NUM\_ROWS\_PRODUCED\_LIMIT query option to limit the number of rows produced by a query. This can be set as the default for user pools where exploratory queries are expected to run and the user might forget to add a LIMIT clause. See [NUM\\_ROWS\\_PRODUCED\\_LIMIT Query Option](#) for more information.
- When using the impala-shell, use the -B option to avoid “prettyprinting” the result set and redirect query results to a file using the --output\_file option. For example:

```
impala-shell --ssl -k -i impala-coordinator1.company.com:21000 -B --output_file /tmp/result
```

## Join query performance tuning

Join queries are tuned by using the join order and using the appropriate type of join.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

### Join order

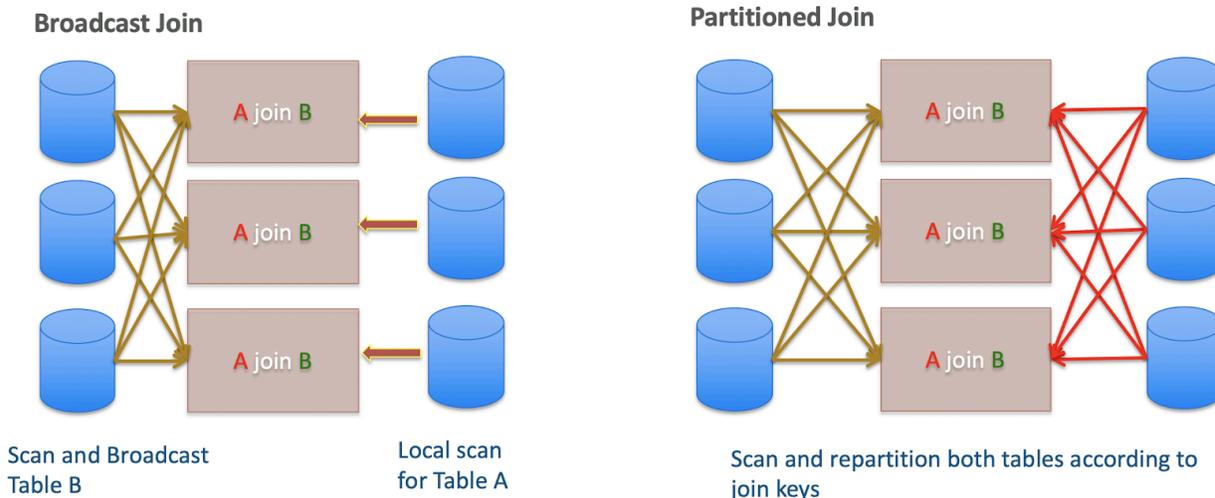
- Make sure that table and column statistics are collected on join columns. This allows Impala to create the most optimal join plan. See [Manually Setting Table and Column Statistics with ALTER TABLE](#) for more information.
- If table or column statistics are not available, join the largest table first. If table or column statistics are not available for some tables in a join, Impala reorders the tables. Impala places tables with statistics on the left side of the join order in descending order of cost, based on overall size and cardinality. Tables without statistics are treated as “zero-size,” which means they are always placed on the right side of the join order.
- If you want to override the default Impala behaviour, use the STRAIGHT\_JOIN technique explained in [Overriding Join Reordering with STRAIGHT\\_JOIN](#).

### Join types

- Broadcast joins are the default join type. In this join type the right-hand table is considered to be smaller than the left-hand table, and its contents are sent to all the other nodes involved in the query.
- Partitioned joins are more suitable for large tables of roughly equal size. With this technique, portions of each table are sent to other appropriate nodes where those subsets of rows can be processed in parallel.
- The choice of broadcast or partitioned joins also depends on statistics being available for all tables in the join.
- Join types can be set manually using the following query options:
  - SET DEFAULT\_JOIN\_DISTRIBUTION\_MODE=shuffle;
  - SET DEFAULT\_JOIN\_DISTRIBUTION\_MODE=broadcast;

See [DEFAULT\\_JOIN\\_DISTRIBUTION\\_MODE Query Option](#) for more information.

The following diagram illustrates the broadcast and partitioned join types:



## Query profiles

When Impala executes any query, it captures the runtime details of the execution in a query profile.

By: [Manish Maheshwari](#), Data Architect and Data Scientist at Cloudera, Inc.

You can view query profiles from the command line by logging in to the Impala shell and running the profile; command. If you want to use Cloudera Manager, from the home page of the Admin Console, select ImpalaQueries . The query profiles display on the Results tab of this page:

Cluster 1

IMPALA-1

Search for Impala queries, e.g. 'rows\_produced > 1000', or press space to start typeahead.

query profiles

| Time                                    | User  | Database | Coordinator   | Duration | Query Type |
|---|-------|----------|---------------|----------|------------|
| 11/21/2019 2:49 PM - 11/21/2019 2:49 PM | admin | default  | :cloudera.com | 8ms      | DDL        |
| 11/21/2019 2:49 PM - 11/21/2019 2:49 PM | admin | default  | :cloudera.com | 777ms    | DDL        |
| 11/21/2019 2:49 PM - 11/21/2019 2:49 PM | admin | default  | :cloudera.com | 12ms     | DDL        |
| 11/21/2019 2:49 PM - 11/21/2019 2:49 PM | admin | default  | :cloudera.com | 16ms     | DDL        |
| 11/21/2019 2:49 PM - 11/21/2019 2:49 PM | admin | default  | :cloudera.com | 166ms    | DDL        |

## Execution summary

Query profiles have an execution summary that shows all the operators involved in the query, the number of nodes on which the operators run on, the time taken, rows produced, and the memory needed on each node.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

The summary section looks like the following example summary for a simple query:

| Operator        | #Hosts | Avg Time  | Max Time  | #Rows   | Est. #Rows | Peak Mem  | Est. Peak Mem | Detail                       |
|-----------------|--------|-----------|-----------|---------|------------|-----------|---------------|------------------------------|
| 10: SORT        | 23     | 8m15s     | 1h        | 290.01M | 44.89M     | 58.29 GB  | 436.00 MB     |                              |
| 05: HASH JOIN   | 23     | 1s733ms   | 9s700ms   | 290.01M | 44.89M     | 20.81 MB  | 1.94 MB       | LEFT SEMI JOIN, BROADCAST    |
| 1--09: EXCHANGE | 23     | 151.541us | 664.160us | 13      | 97         | 112.00 KB | 0             | BROADCAST                    |
| 1 08: AGGREGATE | 23     | 2.731ms   | 11.367ms  | 13      | 97         | 1.97 MB   | 10.00 MB      | FINALIZE                     |
| 1 07: EXCHANGE  | 23     | 319.697us | 1.677ms   | 4.04K   | 968        | 192.00 KB | 0             | HASH(vpmin)                  |
| 1 03: AGGREGATE | 23     | 8s782ms   | 1m11s     | 4.04K   | 968        | 14.63 MB  | 10.00 MB      | STREAMING                    |
| 1 02: SCAN HDFS | 23     | 10s745ms  | 3m41s     | 438.93M | 447.94M    | 1.59 GB   | 4.81 GB       | test.ext_call_event_fact cef |
| 04: HASH JOIN   | 23     | 15s968ms  | 3m20s     | 438.93M | 447.94M    | 13.18 MB  | 1.94 MB       | INNER JOIN, BROADCAST        |
| 1--06: EXCHANGE | 23     | 339.291us | 3.671ms   | 8.04K   | 8.04K      | 544.00 KB | 0             | BROADCAST                    |
| 1 01: SCAN HDFS | 1      | 4.194ms   | 4.194ms   | 8.04K   | 8.04K      | 1.25 MB   | 32.00 MB      | test.date_dim dd             |
| 00: SCAN HDFS   | 23     | 541.093ms | 3s665ms   | 438.93M | 447.94M    | 2.08 GB   | 4.81 GB       | test.ext_call_event_fact cef |

## Query timeline

The query timeline shows the end-to-end time required for a query to execute.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

It captures the detailed time required for each phase of the query. The following image shows a timeline for an example query:

```

Query Compilation: 1m
- Metadata load started: 351.060us (351.060us)
- Metadata load finished. loaded-tables=1/1 load-requests=2 catalog-updates=37: 1m (1m)
- Analysis finished: 1m (1.596ms)
- Value transfer graph computed: 1m (18.009us)
- Single node plan created: 1m (190.919us)
- Distributed plan created: 1m (70.456us)
- Lineage info computed: 1m (38.695us)
- Planning finished: 1m (329.095us)
Query Timeline: 1m2s
- Query submitted: 120.423us (120.423us)
- Planning finished: 1m (1m)
- Submit for admission: 1m (1.758ms)
- Completed admission: 1m (137.653us)
- Ready to start on 1 backends: 1m (438.325us)
- All 1 execution backends (1 fragment instances) started: 1m (2.886ms)
- Rows available: 1m (5.431ms)
- First row fetched: 1m1s (913.202ms)
- Last row fetched: 1m1s (347.012ms)
- Released admission control resources: 1m1s (99.751us)
- Unregister query: 1m2s (560.234ms)
- ComputeScanRangeAssignmentTimer: 1.063ms
ImpalaServer:
- ClientFetchWaitTimer: 1s819ms
- RowMaterializationTimer: 1.133ms

```

# Common scenarios for debugging queries using query profiles

The following topics describe common problems that impact Impala query performance and how to resolve them.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

## Memory limit exceeded

The memory limit exceeded error typically occurs when no memory limit has been set.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

### Error:

```
memory limit exceeded. Limit=2.00 GB Reservation=1.25 GB ReservationLimit=1.60 GB OtherMemory=775.70 MB Total=2.00 GB Peak=2.00 GB
```

### Description and cause:

Typically this occurs because the memory limit is not set for the pool or table statistics are missing, or both.

### Solution:

- Run COMPUTE STATS for each table involved in the query.
- Rerun the query with a memory limit set using the SET MEM\_LIMIT query option. For example:

```
SET MEM_LIMIT=3gb;
```

For more information, see [COMPUTE STATS Statement](#) and [MEM\\_LIMIT Query Option](#).

## Query runs slowly

Impala queries can run slowly for a number of reasons, which are explained in this topic.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

The top reasons that cause queries to run slowly are:

### Missing load

Error:

```
Metadata load finished. loaded-tables=1/1 load-requests=1 catalog-updates=3: 2.75s (2746369188
```

Description and cause:

If Impala does not have the metadata of a table cached in the catalog daemon (catalogd), queries run slowly

Solution:

To avoid these situations, make sure that tables are refreshed in ETL pipelines and that you are using the on-demand metadata feature described in [On-demand metadata and metadata management](#).

### Missing statistics

Error:

```
WARNING: The following tables are missing relevant table and/or column statistics. Default.web_logs
```

Description and cause:

Missing statistics cause wrong join types, for example a hash join rather than a broadcast join. It can also cause wrong join order. Both conditions cause queries to run slower than optimal.

Solution:

Run COMPUTE STATS for each table involved in the query and then rerun the query. See [COMPUTE STATS Statement](#) for more information.

## Admission control

Increasing memory for resource pools can increase query concurrency that is under admission control.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

Impala admission control controls the number of concurrent queries running on the Impala service at any time. If the time difference between the query being queued and when the admission completes is high, then revisit the total memory allocated to that pool:

```
Queued: 127ms (127000586)
Completed admission: 3.50s (3498016148)
```

By increasing the memory, more queries can be run concurrently as long as the total concurrency is within the limits as recommended in [Admission Control](#).

## Client fetch wait timer

The client fetch wait timer determines how long a query is kept open waiting for a client action. Tuning the timer settings as described here can improve Impala performance.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

### Description and cause:

If the client, such as Hue, keeps the query open for pagination, the query is kept running during this time although admission control resources are released. This situation generates a message similar to the following:

```
Rows available: 1.1m (65504301365)
First row fetched: 1.1m (66268304867)
Unregister query: 4.1h (14710907024740)
ImpalaServer
- ClientFetchWaitTimer: 4.1h (14622894717858)
```

### Solution:

Make sure that Hue and Impala close queries by setting the following properties in Cloudera Manager:

- In the Hue Service Advanced Configuration Snippet (Safety Valve) for hue\_safety\_valve.ini configuration property add the following settings:

```
[impala]
set close_queries=true
query_timeout_s=<number_of_seconds>
session_timeout_s=<number_of_seconds>
```

- In the Impala service Idle Session Timeout configuration property, add the number of days, hours, minutes, or seconds that you want sessions to remain idle before they are cancelled.

- In the Impala service Idle Query Timeout configuration property, add the number of days, hours, minutes, or seconds that you want queries to remain idle before they are cancelled.

Do not forget to restart the services in Cloudera Manager after you change the configuration properties.

## Wrong join order

Adjusting query join order or join type can improve Impala performance.

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

Description and cause:

A slow query can also be caused by using incorrect join order or incorrect join types.

Solution:

- To identify the join type and join ordering errors, compare the actual number of rows (#Rows) to the estimated number of rows (Est. #Rows) in the query profile. If the actual versus estimated numbers of rows vary significantly, the join order and the selected join type might be incorrect.
- Ensure that you have statistics collected on all tables and columns so the correct join order and type can be automatically selected.

## Time and data skews

Skews in Avg Time and Max Time, which are listed in the Exec Summary section of the query profile, are possible due to either data skews or workload skews:

By: [Manish Maheshwari, Data Architect and Data Scientist at Cloudera, Inc.](#)

- Data skews can be fixed by running the HDFS balancer periodically and then running `INVALIDATE METADATA` to rebalance the data. See [HDFS Balancers](#) and [INVALIDATE METADATA Statement](#) for more information.
- Workload skews can be addressed by ensuring that similar CPU and memory configurations are used on all the Impala daemons. To address CPU usage, ensure that similar hardware is used on nodes running Impala daemons. To address memory configurations use Cloudera Manager. From its home page select `ImpalaConfiguration`. Then search for the Impala Daemon Memory Limit configuration property:

The screenshot shows the Cloudera Manager interface for Cluster 1. The left sidebar contains navigation options like Clusters, Hosts, Diagnostics, Audits, Charts, Replication, and Administration. The main content area shows the configuration for the IMPALA-1 service. A search bar at the top of the configuration page contains the text "Impala Daemon Memory Limit". Below the search bar, there are filters for SCOPE and CATEGORY. The SCOPE filter shows "Impala Daemon" selected with a count of 3. The CATEGORY filter shows "Advanced" selected with a count of 0. The configuration table shows the following settings:

| Property Name   | Value       | Unit          |
|---|-------------|---------------|
| Cgroup Memory Soft Limit (memory.soft_limit_in_bytes) | -1          | MiB           |
| Cgroup Memory Hard Limit (memory.limit_in_bytes)      | -1          | MiB           |
| Impala Daemon Memory Limit (mem_limit)                | 13110811033 | B ≈ 12.21 GiB |

See [Using the Query Profile for Performance Tuning](#) for more information.