Cloudera Runtime 7.2.6

Managing Apache Impala

Date published: 2019-11-01
Date modified: 2021-01-01

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

READ Support for FULL ACID ORC Tables

FULL ACID v2 transactional tables are readable in Impala without modifying any configurations. You must have Cloudera Runtime 7.2.2 or higher and have connection to Hive Metastore server in order to READ from FULL ACID tables.

There are two types of transactional tables available with Hive ACID.

- INSERT-ONLY
- FULL ACID

Until this release, Impala in CDP supported INSERT-ONLY transactional tables allowing both READ and WRITE operations. The latest version of Impala in CDP now also supports READ of FULL ACID ORC tables.

By default tables created in Impala are INSERT-ONLY managed tables whereas the default tables in Hive are managed tables that are FULL-ACID.

Limitations

- Impala cannot CREATE or WRITE to FULL ACID transactional tables yet. You can CREATE and WRITE FULL ACID transactional tables with transaction scope at the row level via HIVE and use Impala to READ these tables.
- Impala does not support ACID v1.

Concepts Used in FULL ACID v2 Tables

Before beginning to use FULL ACID v2 tables you must be aware of these new concepts like transactions, WriteIds, rowIDs, delta delete directories, locks, etc. that are added to FULL ACID tables to achieve ACID semantics.

Write IDs

For every transaction, both read and write, Hive will assign a globally unique ID. For transactional writes like INSERT and DELETE, it will also assign a table-wise unique ID, a write ID. The write ID range will be encoded in the delta and delete directory names. Results of a DML transactional query are allocated to a location under partition/ table. This location is derived by Write ID allocated to the transaction. This provides Isolation of DML queries and such queries can run in parallel without interfering with each other.

New Sub-directories

New data files resulting from a DML query are written to a unique location derived from WriteId of the transaction. You can find the results of an INSERT query in delta directories under partition/table location. Depending on the operation type there can be two types of delta directories:

- Delta Directory: This type is created for the results of INSERT statements and is named delta_<writeId>_<writeId> under partition/table location.
- Delete Delta Directory: This delta directory is created for results of DELETE statements and is named delete_delta_<writeId>_<writeId> under partition/table location.

UPDATE operations create both delete and delta directories.

Row IDs

rowId is the auto-generated unique ID within the transaction and bucket. This is added to each row to identify each row in a table. RowID is used during a DELETE operation. When a record is deleted from a table, the rowId of the deleted row will be written to the delete_delta directory. So for all subsequent READ operations all rows will be read except these rows.

Schematic differences between INSERT-ONLY and FULL ACID tables

INSERT-ONLY tables do not have a special schema. They store the data just like plain original files from the non-ACID world. However, their files are organized differently. For every INSERT statement the created files are put into a transactional directory which has transactional information in its name.

Full ACID tables do have a special schema. They have row identifiers to support row-level DELETEs. So a row in Full ACID format looks like this:

```
{
   "operation": 0,
   "originalTransaction": 1,
   "bucket": 536870912,
   "rowId": 0,
   "currentTransaction": 1,
   "row": {"i": 1}
   }
```

- The green columns are the hidden/system ACID columns.
- Field “row” holds the user data.
- operation 0 means INSERT, 1 UPDATE, and 2 DELETE. UPDATE will not appear because of the split-update technique (INSERT + DELETE).
- originalTransaction is the write ID of the INSERT operation that created this row.
- bucket is a 32-bit integer defined by BucketCodec class.
- rowId is the auto-generated unique ID within the transaction and bucket.
- currentTransaction is the current write ID. For INSERT, it is the same as currentTransaction. For DELETE, it is the write ID when this record is first created.
- row contains the actual data. For DELETE, row will be null.

Key Differences between INSERT-ONLY and FULL ACID Tables

Before beginning to use FULL ACID v2 tables you must be aware of the key differences between the INSERT-ONLY and FULL-ACID tables.

This table highlights some of the differences between the INSERT-ONLY and FULL ACID tables.

	INSERT-ONLY	FULL ACID
Schema	There is no special data schema. They store the data just like plain original files from the non-ACID world.	Data is in special format, i.e. there are synthetic columns with transactional information in addition to actual data.
Transactional information	Transactional information is encoded in directory names.	Full ACID tables also use the same directory structure as INSERT-only tables. Transactional information is encoded in the directory names. Directory name and filename are the source of transactional information.
Table properties	'transactional'='true', 'transactional_properties'='insert_only'	'transactional'='true'
Supported operations	INSERT-ONLY tables only support insertion of data. UPDATES and DELETES are not supported. These tables also provide CREATE TABLE, DROP TABLE, TRUNCATE, INSERT, SELECT operations.	FULL ACID ORC tables can be READ using IMPALA. These tables also provide UPDATE and DELETE operations at the row level using HIVE. This is achieved using transactions like Insert-Only Tables along with changes in ORC Reader to support deletes.
WRITE operation	WRITE operations are atomic and the results of the insert operation are not visible to other query operations until the operation is committed.	WRITE operations are atomic - The operation either succeeds completely or fails; it does not result in partial data.

	INSERT-ONLY	FULL ACID
INSERT operation	For every INSERT statement the created files are added to a transactional directory which has transactional information in its name.	INSERT operation is done through HIVE and this statement is executed in a single transaction. This operation creates a delta directory containing information about this transaction and its data.
DELETE operation	N/A	DELETE operation is done through HIVE and this event creates a special “delete delta” directory.
UPDATE operation	N/A	UPDATE operation is done through HIVE. This operation is split into an INSERT and DELETE operation. This operation creates a delta dir followed by a delete dir.
READ operation	READ operations always read a consistent snapshot of the data.	READ operations always read a consistent snapshot of the data.
Supported file format	Supports any file formats.	Supports only ORC.
Compactions	Minor and major compactions are supported.	Minor compactions can be created, which means several delta and delete directories can be compacted into one delta and delete directory. Major compactions are also supported.

File structure of FULL ACID transational table

Hive 3 achieves atomicity and isolation of operations on transactional tables by using techniques in write, read, insert, create, delete, and update operations that involve delta files, which can provide query status information and help you troubleshoot query problems. For more information, see Hive 3 ACID transactions

Compaction of Data in FULL ACID Transactional Table

As administrator, you need to manage compaction of delta files that accumulate during data ingestion. Compaction is a process that performs critical housekeeping of files.

Hive creates a set of delta files for each transaction that alters a table or partition and stores them in a separate delta directory. When the number of delta and delete directories in the table grow, the read performance will be impacted, since reading is a process of merging the results of valid transactions. To avoid any compromise on the read performance, occasionally Hive performs compaction, namely minor and major. This process merges these directories while preserving the transaction information.

For more information, see Data Compaction.

Configuring Impala

You must review and configure the mandatory and recommended settings if you installed Impala without Cloudera Manager or if you want to customize your environment after installing Impala. If you installed Impala using Cloudera Manager, some of these configurations are completed automatically.

In some cases, depending on the level of Impala, CDP, and Cloudera Manager, you might need to add particular component configuration details in one of the free-form fields on the Impala configuration pages in Cloudera Manager.

- You must enable short-circuit reads, whether or not Impala was installed through Cloudera Manager. This setting goes in the Impala configuration settings, not the Hadoop-wide settings.
- If you installed Impala in an environment that is not managed by Cloudera Manager, you must enable block location tracking, and you can optionally enable native checksumming for optimal performance.

Short-Circuit Reads

Enabling short-circuit reads allows Impala to read local data directly from the file system. This removes the need to communicate through the DataNodes, improving performance. This setting also minimizes the number of additional copies of data. Short-circuit reads requires libhadoop.so (the Hadoop Native Library) to be accessible to both the server and the client. You must install it from an .rpm, .deb, or parcel to use short-circuit local reads.

Note: If you use Cloudera Manager, you can enable short-circuit reads through a checkbox in the user interface and that setting takes effect for Impala as well.

To Enable Short-Circuit Reads

You can enable short-circuit reads through a checkbox available under Configuration tab for both Impala and HDFS.

1. In Cloudera Manager Clusters select the Impala service, for example, IMPALA.
2. On the Configuration tab, search for dfs.client.read.shortcircuit.
3. Accept the default (enabled), or check to enable the dfs.client.read.shortcircuit property to read the HDFS file blocks directly.
4. Do the above for HDFS service too by clicking Cloudera Manager Clusters and by selecting the HDFS service.
5. Save the changes and Restart the service.

To configure DataNodes for short-circuit reads:

1. On all Impala nodes, configure the following attributes from the HDFS service as shown:
    a. In Cloudera Manager Clusters select the HDFS service, for example, HDFS.
    b. On the Configuration tab, search for dfs.domain.socket.path and set the attribute.
    c. Search for and set the attribute, if necessary, dfs.client.file-block-storage-locations.timeout.millis.
    d. Search for and set the attribute, if necessary, dfs.datanode.hdfs-blocks-metadata.enabled
2. After applying these changes, restart all DataNodes.

Block Location Tracking

Enabling block location metadata allows Impala to know which disk data blocks are located on, allowing better utilization of the underlying disks. Impala will not start unless this setting is enabled.

To enable block location tracking:

1. For each DataNode, enable the following attribute dfs.datanode.hdfs-blocks-metadata.enabled file:
    a. In Cloudera Manager Clusters select the HDFS service, for example, HDFS.
    b. On the Configuration tab, search for dfs.datanode.hdfs-blocks-metadata.enabled and enable the attribute if not already enabled.
2. After applying these changes, restart all DataNodes.

Native Checksumming

Enabling native checksumming causes Impala to use an optimized native library for computing checksums, if that library is available.

To enable native checksumming:

If you installed CDP from packages, the native checksumming library is installed and setup correctly, and no additional steps are required.

If you installed by other means, native checksumming may not be available due to missing shared objects. Finding the message "Unable to load native-hadoop library for your platform... using builtin-java classes where applicable" in the Impala logs indicates native checksumming may be unavailable.

To enable native checksumming, you must build and install libhadoop.so (the Hadoop Native Library).

Modifying Impala Startup Options

You can view and edit the configuration options for the Impala daemons to customize your Impala environment, such as to specify which hosts and ports to use, to assign directories for logging,and to control resource usage and security.

Configuring Impala Startup Options

Navigate to the following page to configure the settings for all the Impala-related daemons:

ClustersImpalaConfiguration.

If the Cloudera Manager interface does not yet have a form field for a newly added option, or if you need to use special options for debugging and troubleshooting, the Advanced category page for each daemon includes one or more Safety Valve fields where you can enter option names directly.

Checking the Values of Impala Startup Options

You can check the current runtime value of all these settings through the Impala Web UI, available by default at:

- http://*IMPALA_HOSTNAME*:25000/varz for the `impalad` daemon
- http://*IMPALA_HOSTNAME*:25010/varz for the `statestored` daemon
- http://*IMPALA_HOSTNAME*:25020/varz for the `catalogd` daemon

Related Information

Web User Interface for Debugging

Monitoring Impala

As an administrator, you monitor Impala service and take action when necessary to keep it running smoothly and avoid conflicts with other components running on the same cluster.

Impala Logs

You can review Impala log files on each host, when you have traced an issue back to a specific system. The Impala logs contain information about any errors Impala encountered, jobs Impala has completed, and settings Impala is configured with.

The logs store information about Impala startup options. This information appears once for each time Impala is started and may include:

- Machine name.
- Impala version number.
- Flags used to start Impala.
- CPU information.
- The number of available disks.

A new set of log files is produced each time the associated daemon is restarted. These log files have long names including a timestamp. The .INFO, .WARNING, and .ERROR files are physically represented as symbolic links to the latest applicable log files.

Review Impala log files on each host, when you have traced an issue back to a specific system:

- By using the web interface at http://host-name:25000/logs where host-name is your Cloudera cluster host name.

  The web interface limits the amount of logging information displayed. To view every log entry, access the log files directly through the file system. Impala log files can often be several megabytes in size.
- By examining the contents of the log file

  By default, the Impala logs are stored at /var/log/impalad/, /var/log/catalogd/, and /var/log/statestore/. The most comprehensive log, showing informational, warning, and error messages, is in the file name impalad.INFO.

  For each of impalad, statestored, catalogd:

  - Examine the .INFO files to see configuration settings for the processes.
  - Examine the .WARNING files to see all kinds of problem information, including such things as suboptimal settings and also serious runtime errors.
  - Examine the .ERROR and/or .FATAL files to see only the most serious errors, if the processes crash, or queries fail to complete. These messages are also in the .WARNING file.

Cloudera Manager collects front-end and back-end logs together into a single view and let you do a search across log data for all the managed nodes in DiagnosticsLogs.

Managing Logs

You must configure the Impala log settings to change the default log locations, to rotate logs, or to log verbose levels.

Procedure

1. To change log file locations:
   a) In Cloudera Manager, navigate to Impala serviceConfiguration.
   b) In the search field, type log_dir.
   c) Specify the new log directories for Impala Daemon, Catalog Server, or StateStore in the respective fields:
      - Impala Daemon Log Directory
      - Catalog Server Log Directory
      - StateStore Log Directory
   d) Click Save Changes and restart Impala.
2. To set up log rotation:
   a) In Cloudera Manager, navigate to Impala serviceConfiguration.
   b) In the search field, type max_log_files.
   c) Specify the values for Impala Daemon, Catalog Server, or StateStore in the respective fields:
      - Impala Daemon Maximum Log Files
      - Catalog Server Maximum Log Files
      - StateStore Maximum Log Files
   d) Click Save Changes and restart Impala.

   The above configuration option specifies how many log files to keep at each severity level (INFO, WARNING, ERROR, and FATAL).

   - A value of 0 preserves all log files, in which case you would set up set up manual log rotation using your Linux tool or technique of choice.
   - A value of 1 preserves only the very latest log file.
   - The default value is 10.

   For some log levels, Impala logs are first temporarily buffered in memory and only written to disk periodically. The --logbufsecs setting controls the maximum time that log messages are buffered for. For example, with the default value of 5 seconds, there may be up to a 5 second delay before a logged message shows up in the log file.

   It is not recommended that you set --logbufsecs to 0 as the setting makes the Impala daemon to spin in the thread that tries to delete old log files.

3. To specify how often the log information is written to disk:
   a) In Cloudera Manager, navigate to Impala serviceConfiguration.
   b) In the search field, type logbuflevel.
   c) Specify the values for Impala Daemon, Catalog Server, or StateStore in the respective fields:

      - Impala Daemon Log Buffer Level
      - Catalog Server Log Buffer Level
      - StateStore Log Buffer Level

      The default is 0, meaning that the log is immediately flushed to disk when Impala outputs an important messages such as a warning or an error, but less important messages such as informational ones are buffered in memory rather than being flushed to disk immediately.
   d) Click Save Changes and restart Impala.

4. To set the logging verbose levels:
   a) In Cloudera Manager, navigate to Impala serviceConfiguration.
   b) In the search field, type GLOG_v.
   c) Specify the values for Impala Daemon, Catalog Server, or StateStore in the respective fields:

      - Impala Daemon Verbose Log Level
      - Catalog Server Verbose Log Level
      - StateStore Verbose Log Level
   d) Click Save Changes and restart Impala.

   As logging levels increase, the categories of information logged are cumulative. For example, GLOG_v=2 records everything GLOG_v=1 records, as well as additional information.

   Increasing logging levels imposes performance overhead and increases log size. Cloudera recommends using GLOG_v=1 for most cases: this level has minimal performance impact but still captures useful troubleshooting information.

   Additional information logged at each level of GLOG_v is as follows:

   - 1: The default level. Logs information about each connection and query that is initiated to an impalad instance, including runtime profiles.
   - 2: Everything from the previous level plus information for each RPC initiated. This level also records query execution progress information, including details on each file that is read.
   - 3: Everything from the previous level plus logging of every row that is read. This level is only applicable for the most serious troubleshooting and tuning scenarios, because it can produce exceptionally large and detailed log files, potentially leading to its own set of performance and capacity problems.

   Note: For performance reasons, Cloudera highly recommends not setting the most verbose logging level to 3.

Impala lineage

You can use the Atlas lineage graph to understand the source and impact of data and changes to data over time and across all your data.

Atlas collects metadata from Impala to represent the lineage among data assets. The Atlas lineage graph shows the input and output processes that the current entity participated in. Entities are included if they were inputs to processes that lead to the current entity or they are output from processes for which the current entity was an input. Impala processes follow this pattern.

Note that lineage is not updated between a table and views that the table is a part of when an Impala ALTER TABLE operation runs on the table.

Related Information

Impala metadata collection

Web User Interface for Debugging

You can use the Impala daemons (`impalad`, `statestored`, and `catalogd`) Web UI to display the diagnostic and status information. Each of these Impala daemons includes a built-in web server.

Impala Daemon Web UI

The Impala Daemon (`impalad`) Web UI includes information about configuration settings, running and completed queries, and associated performance and resource usage for queries. In particular, the Details link for each query displays alternative views of the query including a graphical representation of the plan, and the output of the EXPLAIN, SUMMARY, and PROFILE statements from `impala-shell`. Each host that runs the `impalad` daemon has its own instance of the Web UI, with details about those queries for which that host served as the coordinator. The `impalad` Web UI is primarily used for diagnosing query problems that can be traced to a particular node.

StateStore Web UI

The StateStore (`statestored`) Web UI includes information about memory usage, configuration settings, and ongoing health checks performed by statestored. Because there is only a single instance of the statestored within any Impala cluster, you access the Web UI only on the particular host that serves as the Impala StateStore.

Catalog Server Web UI

The Catalog Server (`catalogd`0 Web UI includes information about the databases, tables, and other objects managed by Impala, in addition to the resource usage and configuration settings of the catalogd. Because there is only a single instance of the catalogd within any Impala cluster, you access the Web UI only on the particular host that serves as the Impala Catalog Server.

Debug Web UI for Impala Daemon

You can use the Impala Daemon (`impalad`) Web UI to view information about configuration settings, running and completed queries, and associated performance and resource usage for queries.

To debug and troubleshoot an `impalad` using a web-based interface, open the URL http ://*IMPALA#SERVER#HOSTNAME*:25000/ in a browser. (For secure clusters, use the prefix https:// instead of http://.)

Because each Impala node produces its own set of debug information, you should choose a specific node that you want to investigate an issue on.

Main Page

The main impalad Web UI page at / lists the following information about the impalad:

- The version of the `impalad` daemon

  The Version section also contains other information, such as when Impala was built and what build flags were used.
- Process start time
- Hardware information
- OS information
- Process information
- CGroup information

Admission Controller Page

The Admission Controller impalad debug Web UI is at /admission page under the main impalad Web UI.

Use the /admission page to troubleshoot queued queries and the admission control.

The admission page provides the following information about each resource pool to which queries have been submitted at least once:

- Time since the statestored received the last update
- A warning if this impalad is considered disconnected from the statestored and thus the information on this page could be stale.
- Pool configuration
- Queued queries submitted to this coordinator, in the order of submission
- Running queries submitted to this coordinator
- Pool stats
    - Average of time in queue: An exponential moving average which represents the average time in queue over the last 10 to 12 queries. If a query is admitted immediately, the wait time of 0 is used in calculating this average wait time.
- Histogram of the distribution of peak memory used by queries admitted to the pool

  Use the histogram to figure out settings for the minimum and maximum query MEM_LIMIT ranges for this pool.

  The histogram displays data for all queries admitted to the pool, including the queries that finished, got canceled, or hit an error.

Click on the pool name to only display information relevant to that pool. You can then refresh the debug page to see only the information for that specific pool.

Click Reset informational stats for all pools to reset the stats that keep track of historical data, such as Totals stats, Time in queue (exponential moving average), and the histogram.

The above information is also available as a JSON object from the following HTTP endpoint:

```
http://IMPALA-SERVER-HOSTNAME:PORT/admission?json
```

Known Backends Page

The Known backends page of the impalad debug Web UI is at /backends under the main impalad Web UI.

This page lists the following info for each of the `impalad` nodes in the cluster. Because each `impalad` daemon knows about every other `impalad` daemon through the StateStore, this information should be the same regardless of which node you select.

- Address of the node: Host name and port
- KRPC address: The KRPC address of the impalad. Use this address when you issue the SHUT DOWN command for this impalad.
- Whether acting as a coordinator
- Whether acting as an executor
- Quiescing status: Specify whether the graceful shutdown process has been initiated on this impa lad.
- Memory limit for admission: The amount of memory that can be admitted to this backend by the admission controller.
- Memory reserved: The amount of memory reserved by queries that are active, either currently executing or finished but not yet closed, on this backend.

  The memory reserved for a query that is currently executing is its memory limit, if set. Otherwise, if the query has no limit or if the query finished executing, the current consumption is used.
- Memory of the queries admitted to this coordinator: The memory submitted to this particular host by the queries admitted by this coordinator.

Catalog Page

The Catalog page of the impalad debug Web UI is at /catalog under the main impalad Web UI.

This page displays a list of databases and associated tables recognized by this instance of `impalad`. You can use this page to locate which database a table is in, check the exact spelling of a database or table name, look for identical table names in multiple databases. The primary debugging use case would be to check if an impalad instance has knowledge of a particular table that someone expects to be in a particular database.

Hadoop Configuration

The Hadoop Configuration page of the impalad debug Web UI is at /hadoop-varz under the main impalad Web UI.

This page displays the Hadoop common configurations that Impala is running with.

JMX

The JMX page of the impalad debug Web UI is at /jmx under the main impalad Web UI.

This page displays monitoring information about various JVM subsystems, such as memory pools, thread management, runtime. etc.

Java Log Level

The Change log level page of the impalad debug Web UI is at /log_level under the main impalad Web UI.

This page displays the current Java and backend log levels, and it allows you to change the log levels dynamically without having to restart the impalad.

Logs Page

The INFO logs page of the impalad debug Web UI is at /logs under the main impalad Web UI.

This page shows the last portion of the impalad.INFO log file, including the info, warning, and error logs for the `impalad`. You can see the details of the most recent operations, whether the operations succeeded or encountered errors. This page provides one central place for the log files and saves you from looking around the filesystem for the log files, which could be in different locations on clusters that use cluster management software.

Memz Page

The Memory Usage page of the impalad debug Web UI is at /memz under the main impalad Web UI.

This page displays the summary and detailed information about memory usage by the impalad.

Metrics Page

The Metrics page of the impalad debug Web UI is at /metrics under the main impalad Web UI.

This page displays the current set of metrics, counters and flags representing various aspects of impalad internal operations.

Queries Page

The Queries page of the impalad debug Web UI is at /queries under the main impalad Web UI.

This page lists:

- Currently running queries
- Queries that have completed their execution, but have not been closed yet
- Completed queries whose details still reside in memory

The queries are listed in reverse chronological order, with the most recent at the top. You can control the amount of memory devoted to completed queries by specifying the -#-#query_log_size startup option for impalad.

This page provides:

- How many SQL statements are failing (State value of EXCEPTION)
- How large the result sets are (# rows fetched)
- How long each statement took (Start Time and End Time)

Click the Details link for a query to display the detailed performance characteristics of that query, such as the profile output.

On the query detail page, in the Profile tab, you have options to export the query profile output to the Thrift, text, or Json format.

The Queries page also includes the Query Locations section that lists the number of running queries with fragments on this host.

RPC Services Page

The RPC durations page of the impalad debug Web UI is at /rpcz under the main impalad Web UI.

This page displays information, such as the duration, about the RPC communications of this impa lad with other Impala daemons.

Sessions Page

The Sessions page of the impalad debug Web UI is at /session under the main impalad Web UI.

This page displays information about the sessions currently connected to this `impalad` instance. For example, sessions could include connections from the impala-shell command, JDBC or ODBC applications, or the Impala Query UI in the Hue web interface.

Threadz Page

The Threads page of the impalad debug Web UI is at /threadz under the main impalad Web UI.

This page displays information about the threads used by this instance of impalad, and it shows which categories they are grouped into. Making use of this information requires substantial knowledge about Impala internals.

Varz Page

The Varz page of the impalad debug Web UI is at /varz under the main impalad Web UI.

This page shows the configuration settings in effect when this instance of impalad communicates with other Hadoop components such as HDFS and YARN. These settings are collected from a set of configuration files.

The bottom of this page also lists all the command-line settings in effect for this instance of impa lad.

Prometheus Metrics Page

At /metrics_prometheus, under the main impalad Web UI, the metrics are generated in Prometheus exposition format that Prometheus can consume for event monitoring and alerting.

The /metrics_prometheus is not shown in the Web UI list of pages.

Debug Web UI for StateStore

You can use the StateStore (`statestored`) Web UI to view information about memory usage, configuration settings, and ongoing health checks performed by statestored.

To debug and troubleshoot the statestored daemon using a web-based interface, open the URL http ://*IMPALA#SERVER#HOSTNAME*:25010/ in a browser. (For secure clusters, use the prefix https:// instead of http://.)

Main Page

The main statestored Web UI page at / lists the following information about the statestored:

- The version of the statestored daemon
- Process start time

- Hardware information
- OS information
- Process information
- CGroup information

Logs Page

The INFO logs page of the debug Web UI is at /logs under the main statestored Web UI.

This page shows the last portion of the statestored.INFO log file, including the info, warning, and error logs for the statestored. You can refer here to see the details of the most recent operations, whether the operations succeeded or encountered errors. This page provides one central place for the log files and saves you from looking around the filesystem for the log files, which could be in different locations on clusters that use cluster management software.

Memz Page

The Memory Usage page of the debug Web UI is at /memz under the main statestored Web UI.

This page displays summary and detailed information about memory usage by the statestored. You can see the memory limit in effect for the node, and how much of that memory Impala is currently using.

Metrics Page

The Metrics page of the debug Web UI is at /metrics under the main statestored Web UI.

This page displays the current set of metrics: counters and flags representing various aspects of stat estored internal operation.

RPC Services Page

The RPC durations page of the statestored debug Web UI is at /rpcz under the main statestored Web UI.

This page displays information, such as the durations, about the RPC communications of this stat estored with other Impala daemons.

Subscribers Page

The Subscribers page of the debug Web UI is at /subscribers under the main statestored Web UI.

This page displays information about the other Impala daemons that have registered with the stat estored to receive and send updates.

Threadz Page

The Threads page of the debug Web UI is at /threadz under the main statestored Web UI.

This page displays information about the threads used by this instance of statestored, and shows which categories they are grouped into. Making use of this information requires substantial knowledge about Impala internals.

Topics Page

The Topics page of the debug Web UI is at /topics under the main statestored Web UI.

This page displays information about the topics to which the other Impala daemons have registered to receive updates.

Varz Page

The Varz page of the debug Web UI is at /varz under the main statestored Web UI.

This page shows the configuration settings in effect when this instance of statestored communicates with other Hadoop components such as HDFS and YARN. These settings are collected from a set of configuration files.

The bottom of this page also lists all the command-line settings in effect for this instance of statesto red.

Prometheus Metrics Page

At /metrics_prometheus, under the main statestored Web UI, the metrics are generated in Prometheus exposition format that Prometheus can consume for event monitoring and alerting.

The /metrics_prometheus is not shown in the Web UI list of pages.

Debug Web UI for Catalog Server

You can use the Catalog Server (`catalogd`) Web UI to view information about the databases, tables, and other objects managed by Impala, in addition to the resource usage and configuration settings of the catalogd.

The main page of the debug Web UI is at http://*IMPALA#SERVER#HOSTNAME*:25020/ (non-secure cluster) or http s://*IMPALA#SERVER#HOSTNAME*:25020/ (secure cluster).

Main Page

The main catalogd Web UI page at / lists the following information about the catalogd:

- The version of the catalogd daemon
- Process start time
- Hardware information
- OS information
- Process information
- CGroup information

Catalog Page

The Catalog page of the debug Web UI is at /catalog under the main catalogd Web UI.

This page displays a list of databases and associated tables recognized by this instance of catalogd. You can use this page to locate which database a table is in, check the exact spelling of a database or table name, look for identical table names in multiple databases. The catalog information is represented as the underlying Thrift data structures.

JMX

The JMX page of the catalogd debug Web UI is at /jmx under the main catalogd Web UI.

This page displays monitoring information about various JVM subsystems, such as memory pools, thread management, runtime. etc.

Java Log Level

The Change log level page of the catalogd debug Web UI is at /log_level under the main catalogd Web UI.

The page displays the current Java and backend log levels and allows you to change the log levels dynamically without having to restart the catalogd

Logs Page

The INFO logs page of the debug Web UI is at /logs under the main catalogd Web UI.

This page shows the last portion of the catalogd.INFO log file, including the info, warning, and error logs for the catalogd daemon. You can refer here to see the details of the most recent operations, whether the operations succeeded or encountered errors. This page provides one central place for the log files and saves you from looking around the filesystem for the log files, which could be in different locations on clusters that use cluster management software.

Memz Page

The Memory Usage page of the debug Web UI is at /memz under the main catalogd Web UI.

This page displays summary and detailed information about memory usage by the catalogd. You can see the memory limit in effect for the node, and how much of that memory Impala is currently using.

Metrics Page

The Metrics page of the debug Web UI is at /metrics under the main catalogd Web UI.

This page displays the current set of metrics: counters and flags representing various aspects of cata logd internal operation.

RPC Services Page

The RPC durations page of the catalogd debug Web UI is at /rpcz under the main catalogd Web UI.

This page displays information, such as the durations, about the RPC communications of this cata logd with other Impala daemons.

Threadz Page

The Threads page of the debug Web UI is at /threadz under the main catalogd Web UI.

This page displays information about the threads used by this instance of catalogd, and shows which categories they are grouped into. Making use of this information requires substantial knowledge about Impala internals.

Varz Page

The Varz page of the debug Web UI is at /varz under the main catalogd Web UI.

This page shows the configuration settings in effect when this instance of catalogd communicates with other Hadoop components such as HDFS and YARN. These settings are collected from a set of configuration files.

The bottom of this page also lists all the command-line settings in effect for this instance of cata logd.

Prometheus Metrics Page

At /metrics_prometheus, under the main catalogd Web UI, the metrics are generated in Prometheus exposition format that Prometheus can consume for event monitoring and alerting.

The /metrics_prometheus is not shown in the Web UI list of pages.

Configuring Impala Web UI

As an administrator, you can diagnose issues with each daemon on a particular host, or perform other administrative actions such as cancelling a running query from the built-in web server's UI. The built-in web server is inlcuded within each of the Impala-related daemons. By default, these web servers are enabled. You can turn them off in a high-security configuration where it is not appropriate for users to have access to this kind of monitoring information through a web interface.

Enabling and Disabling Access to Impala Web Servers

By default, these web servers are enabled. You might turn them off in a high-security configuration where it is not appropriate for users to have access to this kind of monitoring information through a web interface.

To enable or disable Impala Web Servers for Web UI in Cloudera Manager:

- Impala Daemon
  1. Navigate to ClustersImpala ServiceConfiguration.
  2. Select ScopeImpala Daemon .
  3. Select CategoryPorts and Addresses.
  4. Select or clear Enable Impala Daemon Web Server.
  5. Click Save Changes, and restart the Impala service.

- Impala StateStore
  1. Navigate to ClustersImpala ServiceConfiguration.
  2. Select ScopeImpala StateStore.
  3. Select CategoryMain.
  4. Select or clear Enable StateStore Web Server.
  5. Click Save Changes, and restart the Impala service.
- Impala Catalog Server
  1. Navigate to ClustersImpala ServiceConfiguration.
  2. Select ScopeImpala Catalog Server.
  3. Select CategoryMain.
  4. Check or clear Enable Catalog Server Web Server.
  5. Click Save Changes, and restart the Impala service.

Configuring Secure Access for Impala Web Servers

Cloudera Manager supports two methods of authentication for secure access to the Impala Catalog Server, Impala Daemon, and StateStore web servers: password-based authentication and SPNEGO authentication.

Authentication for the three types of daemons can be configured independently.

Configuring Password Authentication

1. Navigate to ClustersImpala ServiceConfiguration.
2. Search for "password" using the Search box in the Configuration tab. This should display the password-related properties (Username and Password properties) for the Impala Daemon, StateStore, and Catalog Server. If there are multiple role groups configured for Impala Daemon instances, the search should display all of them.
3. Enter a username and password into these fields.
4. Click Save Changes, and restart the Impala service.

Now when you access the Web UI for the Impala Daemon, StateStore, or Catalog Server, you are asked to log in before access is granted.

Enabling Kerberos HTTP SPNEGO Authentication for Web UI

To provide security through Kerberos, Impala Web UIs support SPNEGO. SPNEGO is a protocol for securing HTTP requests with Kerberos by passing negotiation tokens through HTTP headers.

To enable authorization using SPNEGO in Cloudera Manager:

1. Navigate to ClustersImpala ServiceConfiguration.
2. Select ScopeImpala 1 (Service-Wid).
3. Select the Enable Kerberos Authentication for HTTP Web-Consoles field.

   This setting is effective only Kerberos is enabled for the HDFS service.
4. Click Save Changes, and restart the Impala service.

Configuring TLS/SSL for Web UI

1. Create or obtain an TLS/SSL certificate.
2. Place the certificate, in .pem format, on the hosts where the Impala Catalog Server and StateStore are running, and on each host where an Impala Daemon is running. It can be placed in any location (path) you choose. If all the Impala Daemons are members of the same role group, then the .pem file must have the same path on every host.
3. Navigate to ClustersImpala ServiceConfiguration.
4. Search for "certificate" using the Search box in the Configuration tab. This should display the certificate file location properties for the Impala Catalog Server, Impala Daemon, and StateStore. If there are multiple role groups configured for Impala Daemon instances, the search should display all of them.

5. In the property fields, enter the full path name to the certificate file, the private key file path, and the password for the private key file..
6. Click Save Changes, and restart the Impala service.

Important: If Cloudera Manager cannot find the .pem file on the host for a specific role instance, that role will fail to start.

When you access the Web UI for the Impala Catalog Server, Impala Daemon, and StateStore, https will be used.

Opening Impala Web UIs

Procedure

1. Navigate to ClustersImpala ServiceConfiguration.
2. Open the appropriate Web UI:Select Web UIImpala Catalog Web UI.
   - To open StateStore Web UI, select Web UIImpala StateStore Web UI.
   - To open Catalog Server Web UI, select Web UIImpala Catalog Web UI.
   - To open Impala Daemon Web UI:
     a. Click the Instances tab.
     b. Click an Impala Daemon instance.
     c. Click Impala Daemon Web UI.

Stopping Impala

Explains how to gracefully shut down Impala Daemons by first allowing running queries a specified amount of time to complete the process.

The flow to gracefully shut down an Impala Daemon is as follows:

1. The shutdown is initiated.
2. The grace period starts. The Impala Daemon informs other coordinators not to schedule any new queries on it. This allows queries already scheduled to run on this daemon by other coordinators to start executing.
3. The grace period expires.
4. The Impala Daemon continuously checks if there are no queries or fragments running.
5. If there are no queries or fragments running, it shuts down.
6. Otherwise, when it reaches the IMPALA_GRACEFUL_SHUTDOWN_DEADLINE duration, Impala Daemon shuts down.

Once Cloudera Manager initiates the stop/shutdown command, the Impala Daemon starts up the graceful shutdown process, and the process cannot be reverted. However, if you need to change the hard deadline in Cloudera Manager, you can cancel the shutdown command, change the Impala Daemon Graceful Shutdown, and start the shutdown command again.

Procedure

1. Optionally, set the grace period.
   a) In Cloudera Manager, navigate to Impala ServiceConfigurationScopeImpala Daemon.
   b) In the Impala Daemon Command Line Argument Advanced Configuration Snippet (Safety Valve) field, specify the grace period:

      --shutdown_grace_period_s=<new grace period in seconds>

      The default grace period is 2 minutes.

      It is strongly recommended that you use the default value and not change the setting.
2. Optionally, set the hard deadline after which Impala is shut down regardless of whether queries are still running on it.
   a) In Cloudera Manager, navigate to Impala ServiceConfigurationScopeImpala Daemon.
   b) In the Impala Graceful Shutdown Deadline field, specify the time to wait for running queries.

      The default is 60 minutes.

      If you specify 0, Impala will shutdown immediately without waiting for running queries
3. In Cloudera Manager, navigate to Impala ServiceInstances.
4. Click an Impala Daemon role.
5. Click ActionsImpala Daemon Graceful Shutdown.

Related Information

SHUTDOWN statement

Securing Impala

You must be aware of this set of security features Impala provides to protect your critical and sensitive data.

The Impala security features have several objectives.

- At the basic level, security prevents accidents or mistakes that could disrupt application processing, delete or corrupt data, or reveal data to unauthorized users.
- Advanced security features and practices can harden the system against malicious users trying to gain unauthorized access or perform other disallowed operations.
- The auditing feature provides a way to confirm that no unauthorized access occurred, and detect whether any such attempts were made.

Based on the above objectives, Impala security features are divided into the following categories.

Authentication

How does Impala verify the identity of the user to confirm that they really are allowed to exercise the privileges assigned to that user? Impala relies on the Kerberos subsystem for authentication.

Authorization

Which users are allowed to access which resources, and what operations are they allowed to perform? Impala relies on the open source Ranger project for authorization. By default (when authorization is not enabled), Impala does all read and write operations with the privileges of the impala user, which is suitable for a development/test environment but not for a secure production environment. When authorization is enabled, Impala uses the OS user ID of the user who runs `impala-shell` or other client program, and associates various privileges with each user.

TLS/SSL Encryption

This feature encrypts TLS/SSL network encryption, between Impala and client programs, and between the Impala-related daemons running on different nodes in the cluster.

Auditing

What operations were attempted, and did they succeed or not? This feature provides a way to look back and diagnose whether attempts were made to perform unauthorized operations. You use this information to track down suspicious activity, and to see where changes are needed in authorization policies. The audit data produced by this feature is collected by the Cloudera Manager product and then presented in a user-friendly form by the Cloudera Manager product.

The following are the important guidelines to protect a cluster running Impala against accidents and mistakes, or malicious attackers trying to access sensitive data. See the subsequent topics that describe the security features in detail.

- Secure the root account. The root user can tamper with the `impalad` daemon, read and write the data files in HDFS, log into other user accounts, and access other system services that are beyond the control of Impala.
- Restrict membership in the sudoers list (in the /etc/sudoers file). The users who can run the sudo command can do many of the same things as the root user.
- Ensure the Hadoop ownership and permissions for Impala data files are restricted.

  The Impala authorization feature makes use of the HDFS file ownership and permissions mechanism.

  1. Set up users and assign them to groups at the OS level, corresponding to the different categories of users with different access levels for various databases, tables, and HDFS locations (URIs).
  2. Create the associated Linux users using the `useradd` command if necessary.
  3. Add Linux users to the appropriate groups with the `usermod` command.
- Ensure the Hadoop ownership and permissions for Impala log files are restricted.

  If you issue queries containing sensitive values in the WHERE clause, such as financial account numbers, those values are stored in Impala log files in the Linux file system, and you must secure those files.
- Ensure that the Impala web UI (available by default on port 25000 on each Impala node) is password-protected.
- Create a policy file that specifies which Impala privileges are available to users in particular Hadoop groups (which by default map to Linux OS groups). Create the associated Linux groups using the `groupadd` command if necessary.
- Design your databases, tables, and views with database and table structure to allow policy rules to specify simple, consistent rules.

  For example, if all tables related to an application are inside a single database, you can assign privileges for that database and use the * wildcard for the table name. If you are creating views with different privileges than the underlying base tables, you might put the views in a separate database so that you can use the * wildcard for the database containing the base tables, while specifying the specific names of the individual views.
- Enable authorization for all impalad daemons.

  The authorization feature does not apply to the `statestored` daemon, which has no access to schema objects or data files.
- Set up authentication using Kerberos, to make sure users really are who they say they are.

Related Information

Configuring Impala Web UI

Configuring Impala TLS/SSL

To protect sensitive information being transmitted, Impala supports TLS/SSL network encryption, between Impala and client programs, and between the Impala-related daemons running on different nodes in the cluster.

To configure Impala to listen for Beeswax and HiveServer2 requests on TLS/SSL-secured ports:

1. In Cloudera Manager, select the Impala service from the Clusters drop down.
2. In the Configuration tab, select ScopeImpala (Service-Wide).
3. Select CategorySecurity.

4. Edit the following property fields:

Property	Description
Enable TLS/SSL for Impala	Encrypt communication between clients (like ODBC, JDBC, and the Impala shell) and the Impala daemon using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)).
Impala TLS/SSL Server Certificate File (PEM Format)	Local path to the X509 certificate that identifies the Impala daemon to clients during TLS/SSL connections. This file must be in PEM format.
Impala TLS/SSL Server Private Key File (PEM Format)	Local path to the private key that matches the certificate specified in the Certificate for Clients. This file must be in PEM format.
Impala TLS/SSL Private Key Password	The password for the private key in the Impala TLS/SSL Server Certificate and Private Key file. If left blank, the private key is not protected by a password.
Impala TLS/SSL CA Certificate	The location on disk of the certificate, in PEM format, used to confirm the authenticity of SSL/TLS servers that the Impala daemons might connect to. Because the Impala daemons connect to each other, this should also include the CA certificate used to sign all the SSL/TLS Certificates. Without this parameter, SSL/TLS between Impala daemons will not be enabled.

Using TLS/SSL with Business Intelligence Tools

You can use Kerberos authentication, TLS/SSL encryption, or both to secure connections from JDBC and ODBC applications to Impala.

Configuring TLS/SSL Communication for the Impala Shell

Typically, a client program has corresponding configuration properties in Cloudera Manager to verify that it is connecting to the right server. For example, with SSL enabled for Impala, you use the following options when starting the `impala-shell` interpreter:

- --ssl: enables TLS/SSL for `impala-shell`.
- --ca_cert: the local pathname pointing to the third-party CA certificate, or to a copy of the server certificate for self-signed server certificates.

If --ca_cert is not set, `impala-shell` enables TLS/SSL, but does not validate the server certificate. This is useful for connecting to a known-good Impala that is only running over TLS/SSL, when a copy of the certificate is not available (such as when debugging customer installations).

For `impala-shell` to successfully connect to an Impala cluster that has the minimum allowed TLS/SSL version set to 1.2 (--ssl_minimum_version=tlsv1.2), the cluster that `impala-shell` runs on must have Python version 2.7.9 or higher (or a vendor-provided Python version with the required support. Some vendors patched Python 2.7.5 versions on Red Hat Enterprise Linux 7 and derivatives).

Specifying TLS/SSL Minimum Allowed Version and Ciphers

Depending on your cluster configuration and the security practices in your organization, you might need to restrict the allowed versions of TLS/SSL used by Impala. Older TLS/SSL versions might have vulnerabilities or lack certain features. You can use startup options for the `impalad`, `catalogd`, and `statestored` daemons to specify a minimum allowed version of TLS/SSL.

Specify one of the following values for the --ssl_minimum_version configuration setting:

- tlsv1: Allow any TLS version of 1.0 or higher. This setting is the default when TLS/SSL is enabled.
- tlsv1.1: Allow any TLS version of 1.1 or higher.
- tlsv1.2: Allow any TLS version of 1.2 or higher.

Along with specifying the version, you can also specify the allowed set of TLS ciphers by using the --ssl_cipher_list configuration setting. The argument to this option is a list of keywords, separated by colons, commas, or spaces, and optionally including other notation. For example:

```
--ssl_cipher_list="RC4-SHA,RC4-MD5"
```

By default, the cipher list is empty, and Impala uses the default cipher list for the underlying platform. See the output of `man ciphers` for the full set of keywords and notation allowed in the argument string.

Impala Authentication

Authentication is the mechanism to ensure that only specified hosts and users can connect to Impala.

Authentication feature verifies that when clients connect to Impala, they are connected to a legitimate server. The feature prevents spoofing such as *impersonation* (setting up a phony client system with the same account and group names as a legitimate user) and *man-in-the-middle attacks* (intercepting application requests before they reach Impala and eavesdropping on sensitive information in the requests or the results).

Impala automatically handles both Kerberos and LDAP authentication. Each `impalad` daemon can accept both Kerberos and LDAP requests through the same port. No special actions need to be taken if some users authenticate through Kerberos and some through LDAP.

You can also make proxy connections to Impala through Apache Knox.

Regardless of the authentication mechanism used, Impala always creates HDFS directories and data files owned by the same user (typically impala). Once you are finished setting up authentication, set up authorization to control user-level access to databases, tables, columns, partitions when they connect through Impala.

Configuring Kerberos Authentication

Impala supports an enterprise-grade authentication system called Kerberos. Kerberos provides strong security benefits including capabilities that render intercepted authentication packets unusable by an attacker. It virtually eliminates the threat of impersonation by never sending a user's credentials in cleartext over the network.

Requirements for Using Impala with Kerberos

Important:

- If you plan to use Impala in your cluster, you must configure your KDC to allow tickets to be renewed, and you must configure krb5.conf to request renewable tickets. Typically, you can do this by adding the max_renewable_life setting to your realm in kdc.conf, and by adding the renew_lifetime parameter to the libdefaults section of krb5.conf.

Impala supports the Cloudera ODBC driver and the Kerberos interface provided. To use Kerberos through the ODBC driver, the host type must be set depending on the level of the ODBD driver:

- SecImpala for the ODBC 1.0 driver.
- SecBeeswax for the ODBC 1.2 driver.
- Blank for the ODBC 2.0 driver or higher, when connecting to a secure cluster.
- HS2NoSasl for the ODBC 2.0 driver or higher, when connecting to a non-secure cluster.

Enabling Kerberos in Impala-shell

To enable Kerberos in the Impala shell, start the `impala-shell` command using the -k flag.

Configuring Impala to Support Kerberos Security

Enabling Kerberos authentication for Impala involves steps that can be summarized as follows:

- Creating service principals for Impala and the HTTP service. Principal names take the form: *SERVICENAME/FULLY.QUALIFIED.DOMAIN.NAME@KERBEROS.REALM*
- Creating, merging, and distributing key tab files for these principals.
- Editing /etc/default/impala (in cluster not managed by Cloudera Manager), or editing the Security settings in the Cloudera Manager interface, to accommodate Kerberos authentication.

Enabling Access to Internal Impala APIs for Kerberos Users

For applications that need direct access to Impala APIs, without going through the HiveServer2 or Beeswax interfaces, you can specify a list of Kerberos users who are allowed to call those APIs. By default, the impala and hdfs users are the only ones authorized for this kind of access. Any users not explicitly authorized through the internal _principals_whitelist configuration setting are blocked from accessing the APIs. This setting applies to all the Impala-related daemons, although currently it is primarily used for HDFS to control the behavior of the catalog server.

Mapping Kerberos Principals to Short Names for Impala

Impala can support the additional mapping rules that will be inserted before rules generated from the list of trusted realms and before the default rule. The support is disabled by default in Impala.

To enable mapping Kerberos principals to short names:

1. In Cloudera Manager, select the Impala service.
2. In the Configuration tab, select Impala (Service-Wide) in Scope and Advanced in Category.
3. Select the Use HDFS Rules to Map Kerberos Principals to Short Names field.
4. Click Save Changes, and restart the Impala service.

Configuring LDAP Authentication

Impala uses LDAP for authentication, verifying the credentials of each user who connects through `impala-shell`, Hue, a Business Intelligence tool, JDBC or ODBC applications.

About this task

Authentication is the process of allowing only specified named users to access the server (in this case, the Impala server). This feature is crucial for any production deployment, to prevent misuse, tampering, or excessive load on the server.

Only the connections between clients and Impala can be authenticated by LDAP.

Procedure

To enable and configure LDAP:

1. In Cloudera Manager, select the Impala service.
2. In the Configuration tab, type ldap in the search box. The fields for LDAP configuration will be listed.
3. Set the following fields to enable LDAP.

   Enable LDAP Authentication (enable_ldap_auth)

   Enables LDAP-based authentication between the client and Impala.

   LDAP URL (ldap_uri)

   Sets the URI of the LDAP server to use. Typically, the URI is prefixed with ldap://. You can specify secure SSL-based LDAP transport by using the prefix ldaps://. The URI can optionally specify the port, for example: ldap://ldap_server.example.com:389 or ldaps://ldap_server.example.com:636. (389 and 636 are the default ports for non-SSL and SSL LDAP connections, respectively.)

4. Set the following fields to support custom bind strings.

   When Impala connects to LDAP, it issues a bind call to the LDAP server to authenticate as the connected user. Impala clients, including the Impala-shell, provide the short name of the user to Impala. This is necessary so that Impala can use role-based access, which uses short names.

   However, LDAP servers often require more complex, structured usernames for authentication. Impala supports three ways of transforming the short name (for example, 'henry') to a more complicated string. If necessary, specify one of the following configuration options when starting the `impalad` daemon.

   Active Directory Domain (--ldap_domain)

   Replaces the username with a string *USERNAME@LDAP_DOMAIN*.

   LDAP BaseDN (--ldap_baseDN)

   Replaces the username with a “distinguished name” (DN) of the form: uid=*USERID*,ldap_baseDN. (This is equivalent to a Hive option).

   LDAP Pattern (--ldap_bind_pattern)

   This is the most general option, and replaces the username with the string *LDAP_BIND_PATTERN* where all instances of the string #UID are replaced with *USERID*. For example, an ldap_bind_pa ttern of "user=#UID,OU=foo,CN=bar" with a username of henry will construct a bind name of "use r=henry,OU=foo,CN=bar".

   The above options are mutually exclusive, and Impala does not start if more than one of these options are specified.

5. Set the following fields for secure LDAP connections.

   To avoid sending credentials over the wire in cleartext, you must configure a secure connection between both the client and Impala, and between Impala and the LDAP server. The secure connection could use SSL or TLS.

   Secure LDAP connections through SSL:

   For SSL-enabled LDAP connections, specify a prefix of ldaps:// instead of ldap://. Also, the default port for SSL-enabled LDAP connections is 636 instead of 389.

   Secure LDAP connections through TLS:

   TLS, the successor to the SSL protocol, is supported by most modern LDAP servers. Unlike SSL connections, TLS connections can be made on the same server port as non-TLS connections. To secure all connections using TLS, specify the following flags as startup options to the `impalad` daemon:

   Enable LDAP TLS (--ldap_tls_

   Tells Impala to start a TLS connection to the LDAP server, and to fail authentication if it cannot be done.

   LDAP Server CA Certificate (--ldap_ca_certificate)

   Specifies the location of the certificate in standard .PEM format. Store this certificate on the local filesystem, in a location that only the impala user and other trusted users can read.

6. Click Save Changes and restart the Impala service.

Enabling LDAP in Hue

To connect to Impala using LDAP authentication through Hue, enable the LDAP setting in Hue.

Procedure

1. Go to the Hue service.
2. Click the Configuration tab.
3. Select ScopeHue Server.
4. Select CategoryAdvanced.

5. Add the following properties to the Hue Server Advanced Configuration Snippet (Safety Valve) for hue_safety_valve_server.ini property

```
[impala]
auth_username=<LDAP username of Hue user to be authenticated>
auth_password=<LDAP password of Hue user to be authenticated>
```

6. Click Save Changes.

Enabling LDAP Authentication for impala-shell

To connect to Impala using LDAP authentication, you specify command-line options to the `impala-shell` command interpreter and enter the password when prompted.

-l

Enables LDAP authentication.

-u

Sets the user. Per Active Directory, the user is the short username, not the full LDAP distinguished name. If your LDAP settings include a search base, use the --ldap_bind_pattern on the `impalad` daemon to translate the short user name from `impala-shell` automatically to the fully qualified name.

`impala-shell` automatically prompts for the password.

Impala Authorization

Authorization determines which users are allowed to access which resources, and what operations they are allowed to perform. You use Apache Ranger to enable and manage authorization in Impala.

Supported Ranger Features in Impala

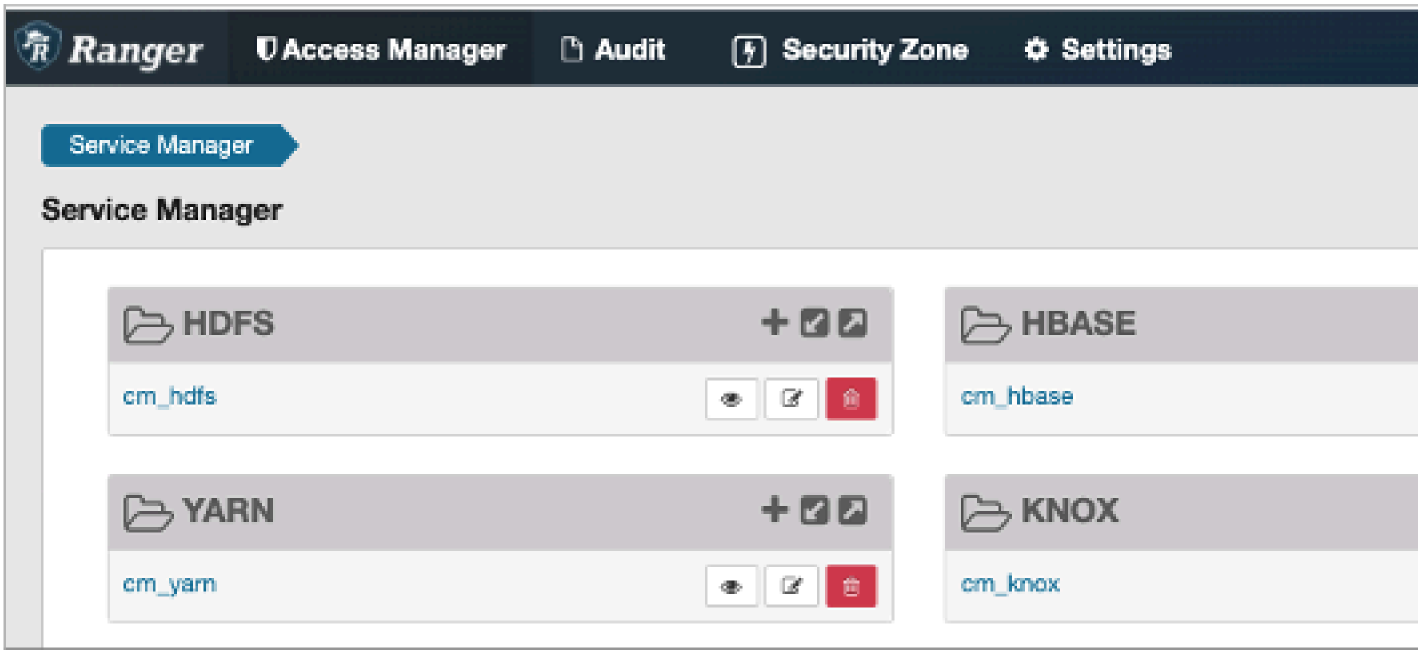
- Resource-based and tag-based authorization policies
- Resource-based and tag-based column masking

Unsupported Ranger Features in Impala

- Row-level filtering

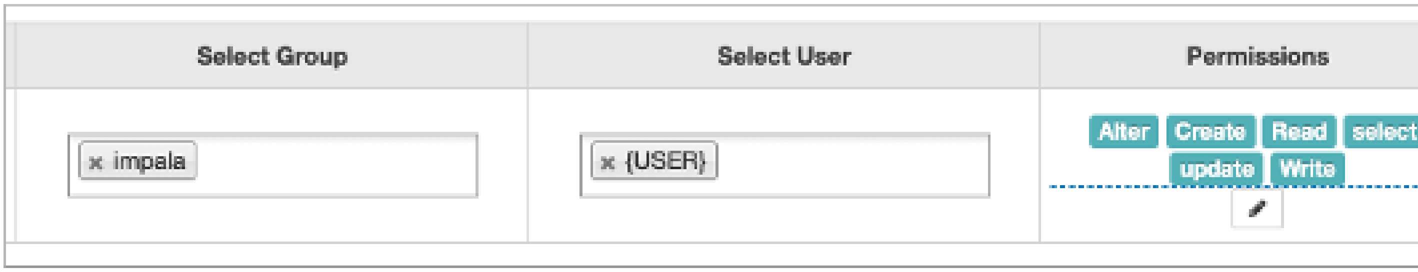
Using Resource-based Authorization Policies for Impala

In the Ranger Service Manager, you can use the Hadoop SQL preloaded resource-based and tag-based services and policies to authorize access to Impala:



You can configure services for Impala as described in Using Ranger to Provide Authorization in CDP.

For example, you can edit the all-global policy for an Impala user:



For information about using tag-based policies, see Tag-based column masking in Hive with Ranger policies.

Privilege Model

You set up privileges through the GRANT and REVOKE statements in either Impala or Hive. Then both components use those same privileges automatically.

By default, when authorization is not enabled, Impala does all read and write operations with the privileges of the impala user, which is suitable for a development/test environment but not for a secure production environment. When authorization is enabled, Impala uses the OS user ID of the user who runs `impala-shell` or other client programs, and associates various privileges with each user.

Privileges can be granted on different resources in the schema and are associated with a level in the resource hierarchy. A privilege on a particular resource automatically inherits the same privilege of its parent.

The resource hierarchy is:

```
Server
    URI
    Database
        Table
            Column
        Function
```

The table-level privileges apply to views as well. Anywhere you specify a table name, you can specify a view name instead.

You can specify privileges for individual columns.

The table below lists the minimum level of privileges and the scope required to execute SQL statements in Impala. The following notations are used:

- The SERVER resource type in Ranger implies all databases, all tables, all columns, all UDFs, and all URLs.
- ANY denotes the CREATE, ALTER, DROP, SELECT, INSERT, or REFRESH privilege.
- ALL privilege denotes the SELECT, INSERT, CREATE, ALTER, DROP, and REFRESH privileges.
- VIEW_METADATA privilege denotes the SELECT, INSERT, or REFRESH privileges.
- The parent levels of the specified scope are implicitly supported. For example, if a privilege is listed with the TABLE scope, the same privilege granted on DATABASE and SERVER will allow the user to execute that specific SQL statement on TABLE.

Note: For an "impala" user to access any managed tables under the warehouse root directory or external tables with directories outside the warehouse root directory, the "impala" user must have HDFS RW access to the underlying paths that are referenced in a query.

For example, to be able to execute CREATE VIEW, you need the CREATE privilege on the database and the SELE CT privilege on the source table.

SQL Statement	Privileges	Object Type / Resource Type
SELECT	SELECT	TABLE
WITH SELECT	SELECT	TABLE
EXPLAIN SELECT	SELECT	TABLE
INSERT	INSERT	TABLE
EXPLAIN INSERT	INSERT	TABLE
TRUNCATE	INSERT	TABLE
LOAD	INSERT	TABLE
	ALL	URI
CREATE DATABASE	CREATE	SERVER
CREATE DATABASE LOCATION	CREATE	SERVER
	ALL	URI
CREATE TABLE	CREATE	DATABASE
CREATE TABLE LIKE	CREATE	DATABASE
	VIEW_METADATA	TABLE
CREATE TABLE AS SELECT	CREATE	DATABASE
	INSERT	DATABASE
	SELECT	TABLE
EXPLAIN CREATE TABLE AS SELECT	CREATE	DATABASE
	INSERT	DATABASE
	SELECT	TABLE
CREATE TABLE LOCATION	CREATE	TABLE
	ALL	URI
CREATE VIEW	CREATE	DATABASE
	SELECT	TABLE

ALTER DATABASE SET OWNER	ALL WITH GRANT	DATABASE
ALTER TABLE	ALL	TABLE
ALTER TABLE SET LOCATION	ALL	TABLE
	ALL	URI
ALTER TABLE RENAME	CREATE	DATABASE
	ALL	TABLE
ALTER TABLE SET OWNER	ALL WITH GRANT	TABLE
ALTER VIEW	ALL	TABLE
	SELECT	TABLE
ALTER VIEW RENAME	CREATE	DATABASE
	ALL	TABLE
ALTER VIEW SET OWNER	ALL WITH GRANT	VIEW
DROP DATABASE	ALL	DATABASE
DROP TABLE	ALL	TABLE
DROP VIEW	ALL	TABLE
CREATE FUNCTION	CREATE	DATABASE
	ALL	URI
DROP FUNCTION	ALL	DATABASE
COMPUTE STATS	ALL	TABLE
DROP STATS	ALL	TABLE
INVALIDATE METADATA	REFRESH	SERVER
INVALIDATE METADATA <table>	REFRESH	TABLE
REFRESH <table>	REFRESH	TABLE
REFRESH AUTHORIZATION	REFRESH	SERVER
REFRESH FUNCTIONS	REFRESH	DATABASE
COMMENT ON DATABASE	ALL	DATABASE
COMMENT ON TABLE	ALL	TABLE
COMMENT ON VIEW	ALL	TABLE
COMMENT ON COLUMN	ALL	TABLE
DESCRIBE DATABASE	VIEW_METADATA	DATABASE
DESCRIBE <table/view>	VIEW_METADATA	TABLE
If the user has the SELECT privilege at the COLUMN level, only the columns the user has access will show.	SELECT	COLUMN
USE	ANY	TABLE
SHOW DATABASES	ANY	TABLE
SHOW TABLES	ANY	TABLE
SHOW FUNCTIONS	VIEW_METADATA	DATABASE
SHOW PARTITIONS	VIEW_METADATA	TABLE
SHOW TABLE STATS	VIEW_METADATA	TABLE
SHOW COLUMN STATS	VIEW_METADATA	TABLE

SHOW FILES	VIEW_METADATA	TABLE
SHOW CREATE TABLE	VIEW_METADATA	TABLE
SHOW CREATE VIEW	VIEW_METADATA	TABLE
SHOW CREATE FUNCTION	VIEW_METADATA	DATABASE
SHOW RANGE PARTITIONS (Kudu only)	VIEW_METADATA	TABLE
UPDATE (Kudu only)	ALL	TABLE
EXPLAIN UPDATE (Kudu only)	ALL	TABLE
UPSERT (Kudu only)	ALL	TABLE
WITH UPSERT (Kudu only)	ALL	TABLE
EXPLAIN UPSERT (Kudu only)	ALL	TABLE
DELETE (Kudu only)	ALL	TABLE
EXPLAIN DELETE (Kudu only)	ALL	TABLE

The privileges not listed in the table above will be silently ignored by Impala.

Changing Privileges in Impala

Privileges are managed via the GRANT and REVOKE SQL statements that require the Ranger service enabled.

Privileges can be also managed in Ranger UI. Especially, for attribute-based access control, Ranger UI is required to manage authorization.

Impala authorization policies are listed in the Hive service section in Ranger UI.

REFRESH AUTHORIZATION is not required when you make the changes to privileges within Impala. The changes are automatically propagated.

Changing Privileges from Outside of Impala

If you make a change to privileges in Ranger from outside of Impala, e.g. adding a user, removing a user, modifying privileges, there are two options to propagate the change:

- Use the ranger.plugin.hive.policy.pollIntervalMs property to specify how often to do a Ranger refresh. The property is specified in ranger-impala-security.xml in the conf directory under your Impala home directory.
- Run the REFRESH AUTHORIZATION statement to force a refresh.

Warning: As INVALIDATE METADATA is an expensive operation, you should use it judiciously.

Granting Privileges on URI

URIs represent the file paths you specify as part of statements such as CREATE EXTERNAL TABLE and LOAD DATA. Typically, you specify what look like UNIX paths, but these locations can also be prefixed with hdfs:// to make clear that they are really URIs. To set privileges for a URI, specify the name of a directory, and the privilege applies to all the files in that directory and any directories underneath it.

URIs must start with hdfs://, s3a://, adl://, or file://. If a URI starts with an absolute path, the path will be appended to the default filesystem prefix. For example, if you specify:

```
GRANT ALL ON URI '/tmp';
```

The above statement effectively becomes the following where the default filesystem is HDFS.

```
GRANT ALL ON URI 'hdfs://localhost:20500/tmp';
```

When defining URIs for HDFS, you must also specify the NameNode. For example:

```
GRANT ALL ON URI file:///path/to/dir TO <role>
GRANT ALL ON URI hdfs://namenode:port/path/to/dir TO <role>
```

Warning: Because the NameNode host and port must be specified, it is strongly recommended that you use High Availability (HA). This ensures that the URI will remain constant even if the NameNode changes. For example:

```
GRANT ALL ON URI hdfs://ha-nn-uri/path/to/dir TO <role>
```

Note:

- If a user uses Impala SQL engine to access the resource of the specified URL/URI provided as part of the SQL statement and as an admin if you must deny access permissions to this location for this particular user, you must add the permission "All" in the "HADOOP SQL" policy.
- However, if the user is using Hive as the execution engine and to deny access permission to a location, then you must add both the “Read” and “Write” permissions in the field of ‘Permissions’ for the corresponding deny conditions.
- If the same Ranger policy is shared by both Hive and Impala, then you must add “All”, “Read”, and “Write” to the field of ‘Permissions’ to enforce the policy.

Object Ownership

Object ownership for tables, views and databases is enabled by default in Impala.

To define owner specific privileges, go to Ranger UI and define appropriate policies on the {OWNER} user.

The CREATE statements implicitly make the user running the statement the owner of the object. For example, if *USER A* creates a database, *FOO*, via the CREATE DATABASE statement, *USER A* now owns the *FOO* database and is authorized to perform any operation on the *FOO* database.

An ownership can be transferred to another user or role via the ALTER DATABASE, ALTER TABLE, or ALTER VIEW with the SET OWNER clause.

Note: Currently, due to a known issue (IMPALA-8937), until the ownership information is fully loaded in the coordinator catalog cache, the owner of a table might not be able to see the table when executing the SHOW TABLES statement. The owner can still query the table.

Ranger Column Masking for Impala

Ranger column masking hides sensitive columnar data in Impala query output. For example, you can define a policy that reveals only the first or last four characters of column data. Column masking is enabled by default. To disable column masking, modify the following configuration property in all coordinators:

```
--enable_column_masking=false
```

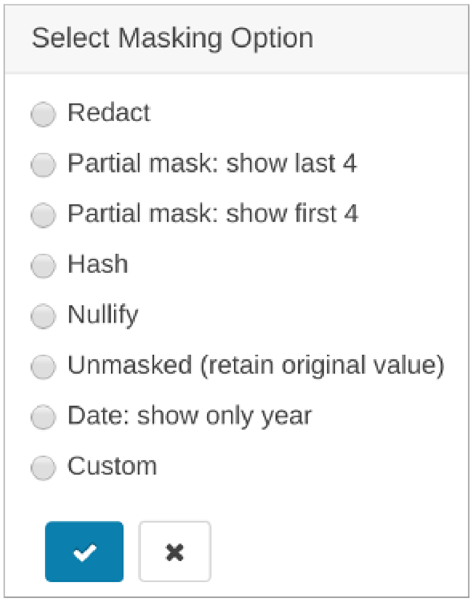
This flag might be removed in a future release. The Impala behavior mimics Hive behavior with respect to column masking.

The following table lists all Impala-supported, built-in mask types for defining column masking in a policy using the Ranger REST API or Ranger UI:

Type	Name	Description	Transformer
MASK	Redact	Replace lowercase with 'x', uppercase with 'X', digits with '0'	mask({col})
MASK_SHOW_LAST_4	Partial mask: show last 4	Show last 4 characters; replace rest with 'x'	mask_show_last_n({col}, 4, 'x', 'x', 'x', -1, '1')

Type	Name	Description	Transformer
MASK_SHOW_FIRST_4	Partial mask: show first 4	Show first 4 characters; replace rest with 'x'	mask_show_first_n({col}, 4, 'x', 'x', 'x', -1, '1')
MASK_HASH	Hash	Hash the value	mask_hash({col})
MASK_NULL	Nullify	Replace with NULL	N/A
MASK_NONE	Unmasked (retain original value)	No masking	N/A
MASK_DATE_SHOW_YEAR	Date: show only year	Date: show only year	mask({col}, 'x', 'x', 'x', -1, '1', 1, 0, -1)
CUSTOM	Custom	Custom	N/A

The table includes the mask name as it appears in the Ranger UI.



Ranger Column Masking Limitations in Impala

- Column masking introduces unused columns during the query analysis and, consequently, additional SELECT privileges checks on all columns of the masked table.
- Impala might produce more than one audit log entry for a column involved in a column masking policy under all of these conditions: the column appears in multiple places in a query, the query is rewritten, or the query is re-analyzed.
- Column masking policies, shared between Hive and Impala, might be affected by SQL/UDF differences between Hive and Impala, as shown in the following example.

  For instance, UTF-8 strings containing non-ASCII characters are not guaranteed to work properly. Suppose a column masking policy masks the last two characters of a string: s => mask_last_n(s, 2, 'x', 'x', 'x', 'x'). Applying this policy, Hive properly masks SQL## to SQLxx, but the Impala masking produces SQL##xx because each Chinese character is encoded in 3 bytes. Impala and Hive do not handle different lengths in the same way.

Limitations on Mask Functions

The mask functions in Hive are implemented through GenericUDFs. Even though Impala users can call Hive UDFs, Impala does not yet support Hive GenericUDFs, so you cannot use Hive's mask functions in Impala. However, Impala has builtin mask functions that are implemented through overloads. In Impala, when using mask functions, not all parameter combinations are supported. These mask functions are introduced in Impala 3.4

The following list includes all the implemented overloads.

- Overloads used by Ranger default masking policies,

- Overloads with simple arguments,
- Overload with all arguments in int type for full functionality. Char argument needs to be converted to their ASCII value.

To list the available overloads, use the following query:

```
show functions in _impala_builtins like "mask*";
```

Note: An error message that states "No matching function with signature: mask..." implies that Impala does not contain the corresponding overload.

Related Information

GRANT statement

REVOKE statement

Resource-based column masking in Hive with Ranger policies

Tag-based column masking in Hive with Ranger policies

Apache Ranger documentation

Configuring Authorization

Enable Ranger authorization in Impala.

Procedure

1. In Cloudera Manager, navigate to the Impala service.
2. In the Configuration tab, type ranger in the search field.
3. Specify the values in the following fields.
    - Ranger Service: Select the Ranger service you are using. This is the server name to use when granting server level privileges.
    - Ranger Policy Cache Directory: Specify the directory where Ranger security policies are cached locally.
    - Ranger DFS Audit Path: Specify the DFS path on which Ranger audits are written. The special placeholder '${ranger_base_audit_url}' should be used as the prefix, in order to use the centralized location defined in the Ranger service.
    - Ranger Audit DFS Spool Dir: Specify the spool directory for Ranger audits being written to DFS.
    - Ranger Audit Solr Spool Dir: Specify the spool directory for Ranger audits being written to Solr.
4. Restart Impala and Hive.

Tuning Impala

The following sections explains procedures for tuning Impala queries and other SQL operations.

Setting Up HDFS Caching

Set up HDFS caching with Impala for improved performance.

Before you begin

Decide how much memory to devote to the HDFS cache on each host. The total memory available for cached data is the sum of the cache sizes on all the hosts. By default, any data block is only cached on one host although you can cache a block across multiple hosts by increasing the replication factor.

Procedure

1. Enable or disable HDFS caching through Cloudera Manager, using the configuration setting Maximum Memory Used for Caching for the HDFS service.

   This control sets the HDFS configuration parameter dfs_datanode_max_locked_memory, which specifies the upper limit of HDFS cache size on each node. Set up the HDFS caching for your Hadoop cluster.

   - All the other manipulation of the HDFS caching settings, such as what files are cached, is done through the command line, either Impala DDL statements or the Linux `hdfs cacheadmin` command.
2. Using the `hdfs cacheadmin` command, set up one or more pools owned by the same user as the `impalad` daemon (typically impala).

   For example:

   ```
   hdfs cacheadmin -addPool four_gig_pool -owner impala -limit 4000000000
   ```
3. Once HDFS caching is enabled and one or more pools are available, on the Impala side, you specify the cache pool name defined by the hdfs cacheadmin command in the Impala DDL statements that enable HDFS caching for a table or partition, such as CREATE TABLE ... CACHED IN *POOL* or ALTER TABLE ... SET CACHED IN *POOL*.
4. You can use `hdfs cacheadmin -listDirectives` to get a list of existing cache pools.
5. You can use `hdfs cacheadmin -listDirectives -stats` to get detailed information about the pools.

Setting up Data Cache for Remote Reads

When Impala compute nodes and its storage are not co-located, the network bandwidth requirement goes up as the network traffic includes the data fetch as well as the shuffling exchange traffic of intermediate results. To mitigate the pressure on the network, you can enable the compute nodes to cache the working set read from remote filesystems, such as, remote HDFS data node, S3, ABFS, ADLS.

About this task

To enable remote data cache as follows.

Procedure

1. In Cloudera Manager, navigate to ClustersImpala Service.
2. In the Configuration tab, select Enable Local Data Cache to enable the local Impala Daemon data cache for caching of remote reads.
3. In Impala Daemon Data Cache Directories, add the directories Impala Daemon will use for caching of remote read data.
4. In Impala Daemon Data Cache Per Directory Capacity, specify the maximum amount of local disk space Impala will use per daemon in each of the configured directrories for caching of remote read data.
5. Click Save Changes and restart the Impala service.

Configuring Dedicated Coordinators and Executors

Configure a dedicated coordinator and a dedicated executor roles to improve scalability of Impala.

Guidelines for Dedicated Coordinators and Executors

- Dedicated coordinator:
  - Should be on an edge node with no other services running on it.
  - Does not need large local disks but still needs some that can be used for Spilling.
  - Require at least the same or even larger memory allocation than executors.

- (Dedicated)Executors:
  - Should be co-located with DataNodes.
  - The number of hosts with dedicated executors typically increases as the cluster grows larger and handles more table partitions, data files, and concurrent queries.

Procedure

1. In Cloudera Manager, navigate to ClustersImpalaConfigurationRole Groups.
2. Click Create to create two role groups with the following values.
   a) Group for Coordinators
      1. Group Name: Coordinators
      2. Role Type: Impala Daemon
      3. Copy from:
         - Select Impala Daemon Default Group if you want the existing configuration gets carried over to the Coordinators.
         - Select None if you want to start with a blank configuration.

   b) Group for Executors
      1. Group Name: Executors
      2. Role Type: Impala Daemon
      3. Copy from:
         - Select Impala Daemon Default Group if you want the existing configuration gets carried over to the Executors.
         - Select None if you want to start with a blank configuration.
3. In the Role Groups page, click Impala Daemon Default Group.
   a) Select the set of nodes intended to be coordinators.
      1. Click Action for Selected and select Move To Different Role Group….
      2. Select the Coordinators.

   b) Select the set of nodes intended to be Executors.
      1. Click Action for Selected and select Move To Different Role Group….
      2. Select the Executors.
4. Click Configuration. In the search field, type Impala Daemon Specialization.
5. Click Edit Individual Values.
6. For Coordinators role group, select COORDINATOR_ONLY.
7. For Executors role group, select EXECUTOR_ONLY.
8. Click Save Changes and then restart the Impala service.

Related Information

Dedicated Coordinator

Managing Resources in Impala

Impala includes the features that balance and maximize resources to improve query performance and scalability of your Cloudera cluster.

A typical deployment uses the following resource management features:

- Static service pools

  Use the static service pools to allocate dedicated resources for Impala to manage and prioritize workloads on clusters.
- Admission control

  Within the constraints of the static service pool, you can further subdivide Impala's resources using dynamic resource pools and admission control.

Related Information

Creating Static Pools

Dynamic Resource Pool Settings

Admission Control and Query Queuing

Admission control is an Impala feature that imposes limits on concurrent SQL queries to avoid resource usage spikes and out-of-memory conditions on busy Cloudera clusters.

The admission control feature lets you set an upper limit on the number of concurrent Impala queries and on the memory used by those queries. Any additional queries are queued until the earlier ones finish, rather than being cancelled or running slowly and causing contention. As other queries finish, the queued queries are allowed to proceed.

You can specify these limits and thresholds for each pool or globally so that you can balance the resource usage and throughput among steady well-defined workloads, rare resource-intensive queries, and ad-hoc exploratory queries.

In addition to the threshold values for currently executing queries, you can place limits on the maximum number of queries that are queued (waiting) and a limit on the amount of time they might wait before returning with an error. These queue settings let you ensure that queries do not wait indefinitely so that you can detect and correct “starvation” scenarios.

Queries, DML statements, and some DDL statements, including CREATE TABLE AS SELECT and COMPUTE STATS are affected by admission control.

On a busy Cloudera cluster, you might find there is an optimal number of Impala queries that run concurrently. For example, when the I/O capacity is fully utilized by I/O-intensive queries, you might not find any throughput benefit in running more concurrent queries. By allowing some queries to run at full speed while others wait, rather than having all queries contend for resources and run slowly, admission control can result in higher overall throughput.

For another example, consider a memory-bound workload such as many large joins or aggregation queries. Each such query could briefly use many gigabytes of memory to process intermediate results. Because Impala by default cancels queries that exceed the specified memory limit, running multiple large-scale queries at once might require re-running some queries that are cancelled. In this case, admission control improves the reliability and stability of the overall workload by only allowing as many concurrent queries as the overall memory of the cluster can accommodate.

Concurrent Queries and Admission Control

One way to limit resource usage through admission control is to set an upper limit on the number of concurrent queries. This is the initial technique you might use when you do not have extensive information about memory usage for your workload. The settings can be specified separately for each dynamic resource pool.

Max Running Queries

Maximum number of concurrently running queries in this pool. The default value is unlimited. (optional)

The maximum number of queries that can run concurrently in this pool. The default value is unlimited. Any queries for this pool that exceed Max Running Queries are added to the admission control queue until other queries finish. You can use Max Running Queries in the early stages of resource management, when you do not have extensive data about query memory usage, to determine if the cluster performs better overall if throttling is applied to Impala queries.

For a workload with many small queries, you typically specify a high value for this setting, or leave the default setting of “unlimited”. For a workload with expensive queries, where some number of concurrent queries saturate the memory, I/O, CPU, or network capacity of the cluster, set the value low enough that the cluster resources are not overcommitted for Impala.

Once you have enabled memory-based admission control using other pool settings, you can still use Max Running Queries as a safeguard. If queries exceed either the total estimated memory or the maximum number of concurrent queries, they are added to the queue.

If Max Running Queries Multiple is set, the Max Running Queries setting is ignored.

Max Running Queries Multiple

This floating point number is multiplied by the current total number of executors at runtime to give the maximum number of concurrently running queries allowed in the pool. The effect of this setting scales with the number of executors in the resource pool.

This calculation is rounded up to the nearest integer, so the result will always be at least one.

If set to zero or a negative number, the setting is ignored.

Max Queued Queries

Maximum number of queries that can be queued in this pool. The default value is 200. (optional).

If Max Queued Queries Multiple is set, the Max Queued Queries setting is ignored.

Max Queued Queries Multiple

This floating point number is multiplied by the current total number of executors at runtime to give the maximum number of queries that can be queued in the pool. The effect of this setting scales with the number of executors in the resource pool.

This calculation is rounded up to the nearest integer, so the result will always be at least one.

If set to zero or a negative number, the setting is ignored.

Queue Timeout

The amount of time, in milliseconds, that a query waits in the admission control queue for this pool before being canceled. The default value is 60,000 milliseconds.

In the following cases, Queue Timeout is not significant, and you can specify a high value to avoid canceling queries unexpectedly:

- In a low-concurrency workload where few or no queries are queued
- In an environment without a strict SLA, where it does not matter if queries occasionally take longer than usual because they are held in admission control

You might also need to increase the value to use Impala with some business intelligence tools that have their own timeout intervals for queries.

In a high-concurrency workload, especially for queries with a tight SLA, long wait times in admission control can cause a serious problem. For example, if a query needs to run in 10 seconds, and you have tuned it so that it runs in 8 seconds, it violates its SLA if it waits in the admission control queue longer than 2 seconds. In a case like this, set a low timeout value and monitor how many queries are cancelled because of timeouts. This technique helps you to discover capacity, tuning, and scaling problems early, and helps avoid wasting resources by running expensive queries that have already missed their SLA.

If you identify some queries that can have a high timeout value, and others that benefit from a low timeout value, you can create separate pools with different values for this setting.

You can combine these settings with the memory-based approach described below. If either the maximum number of or the expected memory usage of the concurrent queries is exceeded, subsequent queries are queued until the concurrent workload falls below the threshold again.

Memory Limits and Admission Control

Each dynamic resource pool can have an upper limit on the cluster-wide memory used by queries executing in that pool.

Use the following settings to manage memory-based admission control.

Max Memory

The maximum amount of aggregate memory available across the cluster to all queries executing in this pool. This should be a portion of the aggregate configured memory for Impala daemons, which will be shown in the settings dialog next to this option for convenience. Setting this to a non-zero value enables memory based admission control.

Impala determines the expected maximum memory used by all queries in the pool and holds back any further queries that would result in Max Memory being exceeded.

If you specify Max Memory, you should specify the amount of memory to allocate to each query in this pool. You can do this in two ways:

- By setting Maximum Query Memory Limit and Minimum Query Memory Limit. This is preferred as it gives Impala flexibility to set aside more memory to queries that are expected to be memory-hungry.
- By setting Default Query Memory Limit to the exact amount of memory that Impala should set aside for queries in that pool.

Note that in the following cases, Impala will rely entirely on memory estimates to determine how much memory to set aside for each query. This is not recommended because it can result in queries not running or being starved for memory if the estimates are inaccurate. And it can affect other queries running on the same node.

- Max Memory, Maximum Query Memory Limit, and Minimum Query Memory Limit are not set, and the MEM_LIMIT query option is not set for the query.
- Default Query Memory Limit is set to 0, and the MEM_LIMIT query option is not set for the query.

If Max Memory Multiple is set, the Max Memory setting is ignored.

Max Memory Multiple

This number of bytes is multiplied by the current total number of executors at runtime to give the maximum memory available across the cluster for the pool. The effect of this setting scales with the number of executors in the resource pool.

If set to zero or a negative number, the setting is ignored.

Minimum Query Memory Limit and Maximum Query Memory Limit

These two options determine the minimum and maximum per-host memory limit that will be chosen by Impala Admission control for queries in this resource pool. If set, Impala admission control will choose a memory limit between the minimum and maximum values based on the per-host memory estimate for the query. The memory limit chosen determines the amount of memory that Impala admission control will set aside for this query on each host that the query is running on. The aggregate memory across all of the hosts that the query is running on is counted against the pool's Max Memory.

Minimum Query Memory Limit must be less than or equal to Maximum Query Memory Limit and Max Memory.

A user can override Impala's choice of memory limit by setting the MEM_LIMIT query option. If the Clamp MEM_LIMIT Query Option setting is set to TRUE and the user sets MEM_LIMIT to a value that is outside of the range specified by these two options, then the effective memory limit will be either the minimum or maximum, depending on whether MEM_LIMIT is lower than or higher the range.

For example, assume a resource pool with the following parameters set:

- Minimum Query Memory Limit = 2GB
- Maximum Query Memory Limit = 10GB

If a user tries to submit a query with the MEM_LIMIT query option set to 14 GB, the following would happen:

- If Clamp MEM_LIMIT Query Option = true, admission controller would override MEM_LIMIT with 10 GB and attempt admission using that value.
- If Clamp MEM_LIMIT Query Option = false, the admission controller will retain the MEM_LIMIT of 14 GB set by the user and will attempt admission using the value.

Clamp MEM_LIMIT Query Option

If this field is not selected, the MEM_LIMIT query option will not be bounded by the Maximum Query Memory Limit and the Minimum Query Memory Limit values specified for this resource pool. By default, this field is selected. The field is disabled if both Minimum Query Memory Limit and Maximum Query Memory Limit are not set.

You can combine the memory-based settings with the upper limit on concurrent queries. If either the maximum number of or the expected memory usage of the concurrent queries is exceeded, subsequent queries are queued until the concurrent workload falls below the threshold again.

Monitoring Admission Control

To see how admission control works for particular queries, examine the profile output or the summary output for the query.

Profile

The information is available through the PROFILE statement in `impala-shell` immediately after running a query in the shell, on the queries page of the Impala debug web UI, or in the Impala log file (basic information at log level 1, more detailed information at log level 2).

The profile output contains details about the admission decision, such as whether the query was queued or not and which resource pool it was assigned to. It also includes the estimated and actual memory usage for the query, so you can fine-tune the configuration for the memory limits of the resource pools.

Summary

The summary output includes the queuing status consisting of whether the query was queued and what was the latest queuing reason.

The information is available in `impala-shell` when the LIVE_PROGRESS or LIVE_SUMMARY query option is set to TRUE.

You can also start an impala-shell session with the --live_progress or --live_summary flags to monitor all queries in that impala-shell session.

Enabling Admission Control

Enable admission control on all production clusters to alleviate possible capacity issues. The capacity issues could be because of a high volume of concurrent queries, because of heavy-duty join and aggregation queries that require large amounts of memory, or because Impala is being used alongside other Hadoop data management components and the resource usage of Impala must be constrained to work well.

About this task

Procedure

1. Navigate to ClustersImpala.
2. In the Configuration tab, navigate to CategoryAdmission Control.
3. Select or clear both the Enable Impala Admission Control and the Enable Dynamic Resource Pools.
4. Enter a Reason for change, and then click Save Changes to commit the changes.
5. Restart the Impala service.
6. After completing this task, for further configuration settings, customize the configuration settings for the dynamic resource pools.

Creating Static Pools

To manage and prioritize workloads on clusters, use the static service pools to allocate dedicated resources for Impala for predictable resource availability. When static service pools are used, Cloudera Manager creates a cgroup in which Impala runs. This cgroup limits memory, CPU and Disk I/O according to the static partitioning policy.

About this task

Create static resource pools for the services running in your Cloudera cluster.

Procedure

1. In Cloudera Manager, navigate to Clusters Static service pools .
2. In the Configuration tab, allocate a portion of resource to each component.
   - HDFS always needs to have a minimum of 5-10% of the resources.
   - Generally, YARN and Impala split the rest of the resources.
     - For mostly batch workloads, you might allocate YARN 60%, Impala 30%, and HDFS 10%.
     - For mostly ad-hoc query workloads, you might allocate Impala 60%, YARN 30%, and HDFS 10%.
3. Click Continue.
4. Review the changes and click Continue.
5. Click Restart Now.

Configuring Dynamic Resource Pool

Admission control and dynamic resource pools are enabled by default. However, until you configure the settings for the dynamic resource pools, the admission control feature is effectively not enabled.

About this task

There is always a resource pool designated as root.default. By default, all Impala queries run in this pool when the dynamic resource pool feature is enabled for Impala. You create additional pools when your workload includes identifiable groups of queries (such as from a particular application, or a particular group within your organization) that have their own requirements for concurrency, memory use, or service level agreement (SLA). Each pool has its own settings related to memory, number of queries, and timeout interval.

Procedure

1. In Cloudera Manager, navigate to ClustersDynamic Resource Pool Configuration. If the cluster has an Impala service, the Resource Pools tab displays under the Impala Admission Control tab.
2. In the Impala Admission Control tab, click Create Resource Pool.

3. Specify a name and resource limits for the pool:
    - In the Resource Pool Name field, type a unique name containing only alphanumeric characters.
    - Optionally, in the Submission Access Control tab, specify which users and groups can submit queries. By default, anyone can submit queries. To restrict this permission, select the Allow these users and groups option and provide a comma-delimited list of users and groups in the Users and Groups fields respectively.
4. Click Create.
5. Click Refresh Dynamic Resource Pools.

Dynamic Resource Pool Settings

Use the following settings to configure your dynamic resource pools for Impala.

Max Memory

Maximum amount of aggregate memory available across the cluster to all queries executing in this pool. This should be a portion of the aggregate configured memory for Impala daemons, which will be shown in the settings dialog next to this option for convenience. Setting this to a non-zero value enables memory based admission control.

Impala determines the expected maximum memory used by all queries in the pool and holds back any further queries that would result in Max Memory being exceeded.

If you specify Max Memory, you should specify the amount of memory to allocate to each query in this pool. You can do this in two ways:

- By setting Maximum Query Memory Limit and Minimum Query Memory Limit. Setting them gives Impala flexibility to set aside more memory to queries that are expected to be memory-hungry.
- By setting Default Query Memory Limit to the exact amount of memory that Impala should set aside for queries in that pool.

Note that if you do not set any of the above options, or set Default Query Memory Limit to 0, Impala will rely entirely on memory estimates to determine how much memory to set aside for each query. This is not recommended because it can result in queries not running or being starved for memory if the estimates are inaccurate.

For example, consider the following scenario:

- The cluster is running impalad daemons on five hosts.
- A dynamic resource pool has Max Memory set to 100 GB.
- The Maximum Query Memory Limit for the pool is 10 GB and Minimum Query Memory Limit is 2 GB. Therefore, any query running in this pool could use up to 50 GB of memory (Maximum Query Memory Limit * number of Impala nodes).
- Impala will execute varying numbers of queries concurrently because queries may be given memory limits anywhere between 2 GB and 10 GB, depending on the estimated memory requirements. For example, Impala may execute up to 10 small queries with 2 GB memory limits or two large queries with 10 GB memory limits because that is what will fit in the 100 GB cluster-wide limit when executing on five hosts.
- The executing queries may use less memory than the per-host memory limit or the Max Memory cluster-wide limit if they do not need that much memory. In general this is not a problem so long as you are able to execute enough queries concurrently to meet your needs.

Minimum Query Memory Limit and Maximum Query Memory Limit

These two options determine the minimum and maximum per-host memory limit that will be chosen by Impala Admission control for queries in this resource pool. If set, Impala admission control will choose a memory limit between the minimum and maximum value based on the per-host memory estimate for the query. The memory limit chosen determines the amount of memory that Impala admission control will set aside for this query on each host that the query is running on. The aggregate memory across all of the hosts that the query is running on is counted against the pool's Max Memory.

Minimum Query Memory Limit must be less than or equal to Maximum Query Memory Limit and Max Memory.

You can override Impala's choice of memory limit by setting the MEM_LIMIT query option. If the Clamp MEM_LIMIT Query Option is selected and the user sets MEM_LIMIT to a value that is outside of the range specified by these two options, then the effective memory limit will be either the minimum or maximum, depending on whether MEM_LIMIT is lower than or higher than the range.

Max Running Queries

Maximum number of concurrently running queries in this pool. The default value is unlimited.

The maximum number of queries that can run concurrently in this pool. The default value is unlimited. Any queries for this pool that exceed Max Running Queries are added to the admission control queue until other queries finish. You can use Max Running Queries in the early stages of resource management, when you do not have extensive data about query memory usage, to determine if the cluster performs better overall if throttling is applied to Impala queries.

For a workload with many small queries, you typically specify a high value for this setting, or leave the default setting of "unlimited". For a workload with expensive queries, where some number of concurrent queries saturate the memory, I/O, CPU, or network capacity of the cluster, set the value low enough that the cluster resources are not overcommitted for Impala.

Once you have enabled memory-based admission control using other pool settings, you can still use Max Running Queries as a safeguard. If queries exceed either the total estimated memory or the maximum number of concurrent queries, they are added to the queue.

Max Queued Queries

Maximum number of queries that can be queued in this pool. The default value is 200. (optional)

Queue Timeout

The amount of time, in milliseconds, that a query waits in the admission control queue for this pool before being canceled. The default value is 60,000 milliseconds.

It the following cases, Queue Timeout is not significant, and you can specify a high value to avoid canceling queries unexpectedly:

- In a low-concurrency workload where few or no queries are queued
- In an environment without a strict SLA, where it does not matter if queries occasionally take longer than usual because they are held in admission control

You might also need to increase the value to use Impala with some business intelligence tools that have their own timeout intervals for queries.

In a high-concurrency workload, especially for queries with a tight SLA, long wait times in admission control can cause a serious problem. For example, if a query needs to run in 10 seconds, and you have tuned it so that it runs in 8 seconds, it violates its SLA if it waits in the admission control queue longer than 2 seconds. In a case like this, set a low timeout value and monitor how many queries are cancelled because of timeouts. This technique helps you to discover capacity, tuning, and scaling problems early, and helps avoid wasting resources by running expensive queries that have already missed their SLA.

If you identify some queries that can have a high timeout value, and others that benefit from a low timeout value, you can create separate pools with different values for this setting.

Clamp MEM_LIMIT Query Option

If this field is not selected, the MEM_LIMIT query option will not be bounded by the Maximum Query Memory Limit and the Minimum Query Memory Limit values specified for this resource pool. By default, this field is selected. The field is disabled if both Minimum Query Memory Limit and Maximum Query Memory Limit are not set.

Admission Control Sample Scenario

You can learn about the factors you must consider when allocating Impala's resources and the process you need to follow to set up admission control for the selected workload.

Anne Chang is administrator for an enterprise data hub that runs a number of workloads, including Impala.

Anne has a 20-node cluster that uses Cloudera Manager static partitioning. Because of the heavy Impala workload, Anne needs to make sure Impala gets enough resources. While the best configuration values might not be known in advance, she decides to start by allocating 50% of resources to Impala. Each node has 128 GiB dedicated to each impalad. Impala has 2560 GiB in aggregate that can be shared across the resource pools she creates.

Next, Anne studies the workload in more detail. After some research, she might choose to revisit these initial values for static partitioning.

To figure out how to further allocate Impala's resources, Anne needs to consider the workloads and users, and determine their requirements. There are a few main sources of Impala queries:

- Large reporting queries executed by an external process/tool. These are critical business intelligence queries that are important for business decisions. It is important that they get the resources they need to run. There typically are not many of these queries at a given time.
- Frequent, small queries generated by a web UI. These queries scan a limited amount of data and do not require expensive joins or aggregations. These queries are important, but not as critical, perhaps the client tries resending the query or the end user refreshes the page.
- Occasionally, expert users might run ad-hoc queries. The queries can vary significantly in their resource requirements. While Anne wants a good experience for these users, it is hard to control what they do (for example, submitting inefficient or incorrect queries by mistake). Anne restricts these queries by default and tells users to reach out to her if they need more resources.

To set up admission control for this workload, Anne first runs the workloads independently, so that she can observe the workload's resource usage in Cloudera Manager. If they could not easily be run manually, but had been run in the past, Anne uses the history information from Cloudera Manager. It can be helpful to use other search criteria (for example, user) to isolate queries by workload. Anne uses the Cloudera Manager chart for Per-Node Peak Memory usage to identify the maximum memory requirements for the queries.

From this data, Anne observes the following about the queries in the groups above:

- Large reporting queries use up to 32 GiB per node. There are typically 1 or 2 queries running at a time. On one occasion, she observed that 3 of these queries were running concurrently. Queries can take 3 minutes to complete.
- Web UI-generated queries use between 100 MiB per node to usually less than 4 GiB per node of memory, but occasionally as much as 10 GiB per node. Queries take, on average, 5 seconds, and there can be as many as 140 incoming queries per minute.
- Anne has little data on ad hoc queries, but some are trivial (approximately 100 MiB per node), others join several tables (requiring a few GiB per node), and one user submitted a huge cross join of all tables that used all system resources (that was likely a mistake).

Based on these observations, Anne creates the admission control configuration with the following pools:

XL_Reporting

Property	Value
Max Memory	1280 GiB
Maximum Query Memory Limit	32 GiB
Minimum Query Memory Limit	32 GiB
Max Running Queries	2

Property	Value
Queue Timeout	5 minutes

This pool is for large reporting queries. To support running 2 queries at a time, the pool memory resources are set to 1280 GiB (aggregate cluster memory). This is for 2 queries, each with 32 GiB per node, across 20 nodes. Anne sets the pool's Maximum Query Memory Limit to 32 GiB so that no query uses more than 32 GiB on any given node. She sets Max Running Queries to 2 (though it is not necessary she do so). She increases the pool's queue timeout to 5 minutes in case a third query comes in and has to wait. She does not expect more than 3 concurrent queries, and she does not want them to wait that long anyway, so she does not increase the queue timeout. If the workload increases in the future, she might choose to adjust the configuration or buy more hardware.

HighThroughput_UI

Property	Value
Max Memory	960 GiB (inferred)
Maximum Query Memory Limit	4 GiB
Minimum Query Memory Limit	2 GiB
Max Running Queries	12
Queue Timeout	5 minutes

This pool is used for the small, high throughput queries generated by the web tool. Anne sets the Maximum Query Memory Limit to 4 GiB per node, and sets Max Running Queries to 12. This implies a maximum amount of memory per node used by the queries in this pool: 48 GiB per node (12 queries * 4 GiB per node memory limit).

Notice that Anne does not set the pool memory resources, but does set the pool's Maximum Query Memory Limit. This is intentional: admission control processes queries faster when a pool uses the Max Running Queries limit instead of the peak memory resources.

This should be enough memory for most queries, since only a few go over 4 GiB per node. For those that do require more memory, they can probably still complete with less memory (spilling if necessary). If, on occasion, a query cannot run with this much memory and it fails, Anne might reconsider this configuration later, or perhaps she does not need to worry about a few rare failures from this web UI.

With regard to throughput, since these queries take around 5 seconds and she is allowing 12 concurrent queries, the pool should be able to handle approximately 144 queries per minute, which is enough for the peak maximum expected of 140 queries per minute. In case there is a large burst of queries, Anne wants them to queue. The default maximum size of the queue is already 200, which should be more than large enough. Anne does not need to change it.

Default

Property	Value
Max Memory	320 GiB
Maximum Query Memory Limit	4 GiB
Minimum Query Memory Limit	2 GiB
Max Running Queries	Unlimited
Queue Timeout	60 Seconds

The default pool (which already exists) is a catch all for ad-hoc queries. Anne wants to use the remaining memory not used by the first two pools, 16 GiB per node (XL_Reporting uses 64 GiB per node, High_Throughput_UI uses 48 GiB per node). For the other pools to get the resources they expect, she must still set the Max Memory resources and the Maximum Query Memory Limit. She sets the Max Memory resources to 320 GiB (16 * 20). She sets the Maximum Query Memory Limit to 4 GiB per node for now. That is somewhat arbitrary, but satisfies some of the ad hoc queries she observed. If someone writes a bad query by mistake, she does not actually want it using all the system resources. If a user has a large query to submit, an expert user can override the Maximum Query Memory Limit (up to 16 GiB per node, since that is bound by the pool Max Memory resources). If that is still insufficient for this user's workload, the user should work with Anne to adjust the settings and perhaps create a dedicated pool for the workload.

Cancelling a Query

Various client applications let you interactively cancel queries submitted or monitored through those applications.

Setting a Time Limit on Query Execution

An Impala administrator can set a default value of the EXEC_TIME_LIMIT_S query option for a resource pool. If a user accidentally runs a large query that executes for longer than the limit, it will be automatically terminated after the time limit expires to free up resources.

You can override the default value per query or per session if you do not want to apply the default EXEC_TIME_LI MIT_S value to a specific query or a session.

Interactively Cancelling a Query

- In the Impala Web UI for the `impalad` host (on port 25000 by default), cancel a query: In the /queriestab, click Cancel for a query in the queries in flight list.
- In `impala-shell`, press ^C
- In Hue, click Cancel from the Watch page.

You can manually cancel a query in the Impala Web UI for the `impalad` host (on port 25000 by default):

Using HLL Datasketch Algorithms in Impala

You can use Datasketch algorithms (HLL) for queries that take too long to calculate exact results due to very large data sets (e.g. number of distinct values).

You may use data sketches (HLL algorithms) to generate approximate results that are much faster to retrieve. HLL is an algorithm that gives approximate answers for computing the number of distinct values in a column. The value returned by this algorithm is similar to the result of COUNT(DISTINCT col) and the NDV function integrated with Impala. However, HLL algorithm is much faster than COUNT(DISTINCT col) and the NDV function and is less memory-intensive for columns with high cardinality.

These HLL algorithms create "sketches", which are data structures containing an approximate calculation of some facet of the data and which are compatible regardless of which system reads or writes them. For e.g., This particular algorithm works in two phases. First it creates a sketch from the dataset provided for the algorithm and then as the second step in the process you can read the estimate from the sketch. Using this approach it's also possible to create granular sketches for a specific dataset (e.g. one sketch per partition) and later on use them to provide answers for different count(distinct) queries without reading the actual data as they can also be merged together. Using HLL you can create sketches by Hive and read it with Impala for getting the estimate and vice versa.

Usage Notes

Query results produced by the HLL algorithm are approximate but within well defined error bounds. Because the number of distinct values in a column returned by this HLL algorithm is an estimate, it might not reflect the precise number of different values in the column, especially if the cardinality is very high.

Data Types

The following is a list of currently supported data types of the datasets that these HLL functions are compatible with.

Supported:

1. Numeric Types:

   TINYINT, INT, BIGINT, FLOAT, DOUBLE
2. String types:

   STRING, CHAR, VARCHAR
3. Other types:

   BINARY

Unsupported:

1. Complex types
2. Some of the scalar types: DECIMAL, TIMESTAMP, BOOLEAN, SMALLINT, DATE

File Formats

The currently supported file formats are ORC and Parquet. Since Impala does not yet have read support for materialized views, it is required for Hive to write the sketches into regular tables (i.e. not into a materialized view), using the BINARY type. Then Impala can be used to read these sketches and calculate the estimate as Hive does.

Limitations:

You can write HLL sketches into a text table, however it may not work as the serialized sketches can contain characters that are special for text format.

Internal Configuration

The configuration is hard coded and the level of compression is HLL_4 and the number of buckets in the HLL array is k=12 (2 power of 12).

Using HLL Sketches

This datasketch algorithm works in two phases.

- First it creates a sketch from the dataset provided for the algorithm.
- As the second step in the process the function provides an estimate from the sketch.

The following example shows how to create a sketch and to create an estimate from the sketch.

Creating a sketch

This query creates a sketch using the ds_hll_sketch function from a column date_string_col containing the supported data type.

```
Select ds_hll_sketch(date_string_col) from tableName;
```

This query receives a primitive expression and returns a serialized HLL sketch that is not in a readble format.

Estimating the number of distinct values in a column

The following query returns a cardinality estimate (similarly to ndv()) for that particular column (date_string_col) by using the HLL function ds_hll_estimate.

```
Select ds_hll_estimate(ds_hll_sketch(date_string_col)) from tableName;
```

This query receives a primitive expression (column name) and then creates a sketch from the column, and then passes the sketch to the outer function that gives a cardinality estimate.

```
    +------------------------------------------------+
    | ds_hll_estimate(ds_hll_sketch(date_string_col))|
    +------------------------------------------------+
    | 729                                            |
```

```
+------------------------------------------------+
Fetched 1 row(s) in 0.25s
```

The above result is similar to the results received using the count(distinct col) or the ndv() approach.

```
Select count(distinct date_string_col) from tableName;
+------------------------------------------------+
| count (distinct date_string_col)               |
+------------------------------------------------+
| 730                                            |
+------------------------------------------------+
```

```
Select ndv(distinct date_string_col) from tableName;
+------------------------------------------------+
| ndv (date_string_col)                          |
+------------------------------------------------+
| 736                                            |
+------------------------------------------------+
```

Inserting the derived sketch into a table

You can create a sketch and save it for later use. If you save the sketches to a table or view with the same granularity (e.g. one sketch for each partition in a table) then later you can simply read the sketch and get the estimate almost for free as the real cost here is just the sketch creation.

The following example shows the creation of a sketch for the column (date_string_col) from the table “tableName” and saving the created sketch into another table called sketch_table and later reading the sketch and its estimate from the saved table called sketch_table.

```
   select year, month, ds_hll_estimate(ds_hll_sketch(date_string_col)) from
tableName group by year, month;
   +--------------------------------------------------------------+
   | year | month | ds_hll_estimate(ds_hll_sketch(date_string_col))|
   +------+-------+-----------------------------------------------+
   | 2010 | 5     | 31                                            |
   | 2010 | 11    | 30                                            |
   | 2010 | 6     | 30                                            |
   | 2010 | 2     | 28                                            |
   | 2010 | 10    | 31                                            |
   | 2009 | 11    | 30                                            |
   | 2009 | 7     | 31                                            |
   | 2009 | 10    | 31                                            |
   | 2010 | 11    | 31                                            |
   | 2009 | 7     | 31                                            |
   | 2010 | 10    | 31                                            |
   | 2010 | 8     | 30                                            |
   | 2010 | 5     | 31                                            |
   | 2009 | 7     | 30                                            |
   | 2010 | 4     | 31                                            |
   | 2010 | 12    | 30                                            |
   | 2009 | 9     | 31                                            |
   | 2009 | 8     | 30                                            |
   | 2010 | 6     | 28                                            |
   | 2010 | 3     | 31                                            |
   | 2010 | 4     | 31                                            |
   | 2010 | 2     | 30                                            |
```

```
| 2009 | 1     | 31                                                |
| 2009 | 12    | 30                                                |
+----------------------------------------------------------------+
```

Insert the created sketch into another table:

```
insert into sketch_table select year, month, ds_hll_sketch(date_string_
col) from tableName group by year, month;
```

Verify the number of rows modified using count function in the table:

```
select count(1) from sketch_table;
```

You can also verify it by getting the estimates from the sketches stored in the table using the following query:

```
select year, month, ds_hll_estimate(sketch_col) from sketch_table;
```

Combining Sketches

You can combine sketches to allow complex queries to be computed from a few number of simple sketches.

```
select ds_hll_estimate(ds_hll_union(sketch_col)) from sketch_table;
```

Non-deterministic vs Deterministic Results

Based on the characteristics of a query, DataSketches HLL might return results in a non-deterministic manner. During the cache populating phase it returns exact results but in the approximating phase the algorithm can be sensitive to the order of data it receives. As a result you might see the results differ in consecutive runs but all of them will be within the error range of the algorithm.

Due to the characteristics of the algorithm, and to how it was implemented in Impala, the results might be non-deterministic when the query is executed on one node and the input data set is big enough that the algorithm starts approximating instead of giving exact results.

Inter-opreability between Hive and Impala

From the above examples you will notice that the NDV function is much faster than the Datasketches. But NDV is used only by Impala and not by Hive whereas the Datasketches are implemented in Hive as well as in Impala. In order to maintain the inter-operability between Hive and Impala we recommend to use Datasketches to calculate the estimate. A sketch created by Hive using Datasketches can be fed to Impala to produce an estimate.

Using KLL Datasketch Algorithms in Impala

You can use the stochastic streaming algorithm (KLL) that uses the percentile/quantile functions to statistically analyze the approximate distribution of comparable data from a very large stream.

Use the ds_kll_quantiles() or its inverse functions the Probability Mass Function ds_kll_PMF() and the Cumulative Distribution Function ds_kll_CDF() to obtain the analysis. Query results produced by this KLL algorithm are approximate but within well defined error bounds.

Data Types and File Formats

KLL function currently supports only floats. If you sketch an int data the KLL function converts it into float.

You can sketch data from any file formats. However it is recommended to save the sketches into parquet tables. It is not recommended to save the sketches to text format.

Using KLL Sketches

This datasketch algorithm works in two phases.

- First it creates a sketch from the dataset provided for the algorithm.
- As the second step in the process you can read the estimate from the sketch.

Using this approach you can create granular sketches for a specific dataset (e.g. one sketch per partition) and later use them to provide answers for different quantile queries without reading the actual data as they can also be merged together. Using KLL you can create sketches by Hive and read it with Impala for getting the estimate and vice versa.

Creating a sketch

The following example shows how to create a sketch using the ds_kll_sketch function from a column (float_col) containing the supported data type. This created sketch is a data structure containing an approximate calculation of some facet of the data and which are compatible regardless of which system reads or writes them.

```
select ds_kll_sketch(float_col) from tableName;
```

where ds_kll_sketch() is an aggregate function that receives a float parameter (e.g. a float column of a table) and returns a serialized Apache DataSketches KLL sketch of the input data set wrapped into STRING type. Since the serialized data might contain unsupported characters this query may error out. So another approach to this method is to insert the derived sketch into a table or a view and later use for quantile approximations.

Inserting the derived sketch into a table

If you save the sketches to a table or view with the same granularity (e.g. one sketch for each partition in a table) then later you can simply read the sketch for quantile approximation.

```
insert into table_name select ds_kll_sketch(float_col) from tableName;
```

Debugging the created sketch

To troubleshoot the sketch created by the ds_kll_sketch function, use ds_kll_stringify on the sketch. ds_kll_stringify () receives a string that is a serialized Apache DataSketches sketch and returns its stringified format by invoking the related function on the sketch's interface.

```
select ds_kll_stringify(ds_kll_sketch(float_col)) from tableName;
```

```
    +-------------------------------------------+
    | ds_kll_stringify(ds_kll_sketch(float_col))|
    +-------------------------------------------+
    | ### KLL sketch summary:                   |
    |    K                     : 200            |
    |    min K                 : 200            |
    |    M                     : 8              |
    |    N                     : 100            |
    |    Epsilon               : 1.33%          |
    |    Epsilon PMF           : 1.65%          |
    |    Empty                 : false          |
    |    Estimation mode       : false          |
    |    Levels                : 1              |
    |    Sorted                : false          |
    |    Capacity items        : 200            |
    |    Retained items        : 100            |
```

```
|    Storage bytes          : 432                   |
|    Min value              : 0                     |
|    Max value              : 9                     |
| ### End sketch summary                            |
+---------------------------------------------+
```

Determining the number of datasets sketched

To determine the number of records sketched into this sketch, use ds_kll_n function.

```
select ds_kll_n(ds_kll_sketch(float_col)) from tableName;
```

```
+------------------------------------+
| ds_kll_n(ds_kll_sketch(float_col)) |
+------------------------------------+
| 100                                |
+------------------------------------+
```

Calculate Quantile

This function ds_kll_quantile() function receives two parameters, a STRING parameter that contains a serialized KLL sketch and a DOUBLE (0.5) that represents the rank of the quantile in the range of [0,1]. E.g. rank=0.5 means the approximate median of all the sketched data.

```
+------------------------------------------------+
| ds_kll_quantile(ds_kll_sketch(float_col), 0.5) |
+------------------------------------------------+
| 4.0                                            |
+------------------------------------------------+
```

This query returns a data (4.0) that is bigger than the 50% of the data.

Calculating quantiles

To calculate the quantiles for multiple rank parameters, use the function ds_kll_quantiles_as_string () that is very similar to ds_kll_quantile() but this function receives any number of rank parameters and returns a comma separated string that holds the results for all of the given ranks.

```
   select ds_kll_quantiles_as_string(ds_kll_sketch(float_col), 0.5, 0.6, 1)
from tableName;
```

```
+-------------------------------------------------------------------+
| ds_kll_quantiles_as_string(ds_kll_sketch(float_col), 0.5, 0.6, 1) |
+-------------------------------------------------------------------+
| 4, 5, 9                                                           |
+-------------------------------------------------------------------+
```

Calculate Rank

This rank function ds_kll_rank() receives two parameters, a STRING that represents a serialized DataSketches KLL sketch and a float to provide a probing value in the sketch.

```
select ds_kll_rank(ds_kll_sketch(float_col), 4) from tableName;
```

```
+-------------------------------------------------+
| ds_kll_rank(ds_kll_sketch(float_col), 4)        |
+-------------------------------------------------+
| 0.48                                            |
+-------------------------------------------------+
```

This query returns a DOUBLE that is the rank of the given probing value in the range of [0,1]. E.g. a return value of 0.48 means that the probing value given as parameter is greater than 48% of all the values in the sketch.

Note: This is only an approximate calculation.

Calculate Probability Mass Function (PMF)

This Probabilistic Mass Function (PMF) ds_kll_pmf_as_string()receives a serialized KLL sketch and one or more float values to represent ranges in the sketched values. In the following example, the float values [1, 3, 4, 8] represent the following ranges: (-inf, 1), [1, 3), [3, 4), [4, 8) [8, +inf)

Note: The input values for the ranges have to be unique and monotonically increasing.

```
select ds_kll_pmf_as_string(ds_kll_sketch(float_col), 1, 3, 4, 8) from table
Name
```

```
+-------------------------------------------------------------+
| ds_kll_pmf_as_string(ds_kll_sketch(float_col), 1, 3, 4, 8) |
+-------------------------------------------------------------+
| 0.12, 0.24, 0.12, 0.36, 0.16                                |
+-------------------------------------------------------------+
```

This query returns a comma separated string where each value in the string is a number in the range of [0,1] and shows what percentage of the data is in the particular ranges.

Calculate Cumulative Distribution Function (CDF)

This Cumulative Distribution Function (CDF) ds_kll_cmf_as_string()receives a serialized KLL sketch and one or more float values to represent ranges in the sketched values. In the following example, the float values [1, 3, 4, 8] represents the following ranges: (-inf, 1), (-inf, 3), (-inf, 4), (-inf, 8), (-inf, +inf)

Note: The input values for the ranges have to be unique and monotonically increasing.

```
select ds_kll_cdf_as_string(ds_kll_sketch(float_col), 1, 3, 4, 8) from table
Name;
```

```
    +--------------------------------------------------------------+
    | ds_kll_cdf_as_string(ds_kll_sketch(float_col), 1, 3, 4, 8) |
    +--------------------------------------------------------------+
    | 0.12, 0.36, 0.48, 0.84, 1                                    |
    +--------------------------------------------------------------+
```

This query returns a comma separated string where each value in the string is a number in the range of [0,1] and shows what percentage of the data is in the particular ranges.

Calculate the Union

To take advantage of the UNION function, create granular sketches for a specific dataset (one sketch per partition), write these sketches to a separate table and based on the partition you are interested in, you can UNION the relevant sketches together to get an estimate.

```
insert into sketch_tbl select ds_kll_sketch(float_col) from tableName;
```

```
select ds_kll_quantile(ds_kll_union(sketch_col), 0.5) from sketch_tbl
    where partition_col=1 OR partition_col=5;
```

This function ds_kll_union() receives a set of serialized Apache DataSketches KLL sketches produced by ds_kll_s ketch() and merges them into a single sketch.

Managing Metadata in Impala

This section describes various knobs you can use to control how Impala manages its metadata in order to improve performance and scalability.

On-demand Metadata

With the on-demand metadata feature, the Impala coordinators pull metadata as needed from catalogd and cache it locally. The cached metadata gets evicted automatically under memory pressure.

The granularity of on-demand metadata fetches is at the partition level between the coordinator and catalogd. Common use cases like add/drop partitions do not trigger unnecessary serialization/deserialization of large metadata.

The feature can be used in either of the following modes.

Metadata on-demand mode

In this mode, all coordinators use the metadata on-demand.

Set the following on catalogd:

```
--catalog_topic_mode=minimal
```

Set the following on all impalad coordinators:

```
--use_local_catalog=true
```

Mixed mode

In this mode, only some coordinators are enabled to use the metadata on-demand.

We recommend that you use the mixed mode only for testing local catalog’s impact on heap usage.

Set the following on catalogd:

```
--catalog_topic_mode=mixed
```

Set the following on impalad coordinators with metdadata on-demand:

```
--use_local_catalog=true
```

Limitation:

HDFS caching is not supported in On-demand metadata mode coordinators.

Note:

In Impala 3.4.0 and above, global INVALIDATE METADATA statement is supported when On-demand feature is enabled.

INVALIDATE METADATA Usage Notes:

To return accurate query results, Impala needs to keep the metadata current for the databases and tables queried. Through "automatic invalidation" or "HMS event polling" support, Impala automatically picks up most changes in metadata from the underlying systems. However there are some scenarios where you might need to run INVALIDA TE METADATA or REFRESH.

- if some other entity modifies information used by Impala in the metastore, the information cached by Impala must be updated via INVALIDATE METADATA or REFRESH,
- if you have "local catalog" enabled without "HMS event polling" and need to pick up metadata changes that were done outside of Impala in Hive and other Hive client, such as SparkSQL,
- and so on.
- **Note:** As this is a very expensive operation compared to the incremental metadata update done by the REFRESH statement, when possible, prefer REFRESH rather than INVALIDATE METADATA.

Related Information

Invalidate Metadata Statement

Automatic Invalidation of Metadata Cache

To keep the size of metadata bounded, the Impala Catalog Server periodically scans all the tables and invalidates those not recently used.

There are two types of configurations in Catalog Server that control the automatic invalidation of metadata in the Catalog Server Command Line Argument Advanced Configuration Snippet (Safety Valve) field in Cloudera Manager.

Time-based cache invalidation

Catalogd invalidates tables that are not recently used in the specified time period (in seconds).

The ##invalidate_tables_timeout_s flag needs to be applied to both impalad and catalogd.

Memory-based cache invalidation

When the memory pressure reaches 60% of JVM heap size after a Java garbage collection in cata logd, Impala invalidates 10% of the least recently used tables.

The ##invalidate_tables_on_memory_pressure flag needs to be applied to both impalad and cata logd.

Automatic invalidation of metadata provides more stability with lower chances of running out of memory, but the feature could potentially cause performance issues and may require tuning.

Automatic Invalidation/Refresh of Metadata

In this release, you can invalidate or refresh metadata automatically after changes to databases, tables or partitions render metadata stale. You control the synching of tables or database metadata by basing the process on events. You learn how to access metrics and state information about the invalidate event processor.

When tools such as Hive and Spark are used to process the raw data ingested into Hive tables, new HMS metadata (database, tables, partitions) and filesystem metadata (new files in existing partitions/tables) are generated. In previous versions of Impala, in order to pick up this new information, Impala users needed to manually issue an INVALIDATE or REFRESH commands.

When automatic invalidate/refresh of metadata is enabled,, the Catalog Server polls Hive Metastore (HMS) notification events at a configurable interval and automatically applies the changes to Impala catalog.

Impala Catalog Server polls and processes the following changes.

- Invalidates the tables when it receives the ALTER TABLE event.
- Refreshes the partition when it receives the ALTER, ADD, or DROP partitions.
- Adds the tables or databases when it receives the CREATE TABLE or CREATE DATABASE events.
- Removes the tables from catalogd when it receives the DROP TABLE or DROP DATABASE events.
- Refreshes the table and partitions when it receives the INSERT events.

  If the table is not loaded at the time of processing the INSERT event, the event processor does not need to refresh the table and skips it.
- Changes the database and updates catalogd when it receives the ALTER DATABASE events. The following changes are supported. This event does not invalidate the tables in the database.
  - Change the database properties
  - Change the comment on the database
  - Change the owner of the database
  - Change the default location of the database

    Changing the default location of the database does not move the tables of that database to the new location. Only the new tables which are created subsequently use the default location of the database in case it is not provided in the create table statement.

This feature is controlled by the ##hms_event_polling_interval_s flag. Start the catalogd with the ##hms_event_polling_interval_s flag set to a positive integer to enable the feature and set the polling frequency in seconds. We recommend the value to be less than 5 seconds.

The following use cases are not supported:

- When you bypass HMS and add or remove data into table by adding files directly on the filesystem, HMS does not generate the INSERT event, and the event processor will not invalidate the corresponding table or refresh the corresponding partition.

  It is recommended that you use the LOAD DATA command to do the data load in such cases, so that event processor can act on the events generated by the LOAD command.

- The Spark API that saves data to a specified location does not generate events in HMS, thus is not supported. For example:

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

This feature is turned off by default with the ##hms_event_polling_interval_s flag set to 0.

Disable Event Based Automatic Metadata Sync

When the ##hms_event_polling_interval_s flag is set to a non-zero value for your catalogd, the event-based automatic invalidation is enabled for all databases and tables. If you wish to have the fine-grained control on which tables or databases need to be synced using events, you can use the impala.disableHmsSync property to disable the event processing at the table or database level.

When you add the DBPROPERTIES or TBLPROPERTIES with the impala.disableHmsSync key, the HMS event based sync is turned on or off. The value of the impala.disableHmsSync property determines if the event processing needs to be disabled for a particular table or database.

- If 'impala.disableHmsSync'='true', the events for that table or database are ignored and not synced with HMS.
- If 'impala.disableHmsSync'='false' or if impala.disableHmsSync is not set, the automatic sync with HMS is enabled if the ##hms_event_polling_interval_s global flag is set to non-zero.
- To disable the event based HMS sync for a new database, set the impala.disableHmsSync database properties in Hive as currently, Impala does not support setting database properties:

```
CREATE DATABASE <name> WITH DBPROPERTIES ('impala.disableHmsSync'='true');
```

- To enable or disable the event based HMS sync for a table:

```
CREATE TABLE <name> ... TBLPROPERTIES ('impala.disableHmsSync'='true' |
'false');
```

- To change the event based HMS sync at the table level:

```
ALTER TABLE <name> SET TBLPROPERTIES ('impala.disableHmsSync'='true' | '
false');
```

When both table and database level properties are set, the table level property takes precedence. If the table level property is not set, then the database level property is used to evaluate if the event needs to be processed or not.

If the property is changed from true (meaning events are skipped) to false (meaning events are not skipped), you need to issue a manual INVALIDATE METADATA command to reset event processor because it doesn't know how many events have been skipped in the past and cannot know if the object in the event is the latest. In such a case, the status of the event processor changes to NEEDS_INVALIDATE.

Metrics for Event Based Automatic Metadata Sync

You can use the web UI of the catalogd to check the state of the automatic invalidate event processor.

By default, the debug web UI of catalogd is at http://*IMPALA-SERVER-HOSTNAME*:25020 (non-secure cluster) or https://*IMPALA-SERVER-HOSTNAME*:25020 (secure cluster).

Under the web UI, there are two pages that presents the metrics for HMS event processor that is responsible for the event based automatic metadata sync.

- /metrics#events
- /events

  This provides a detailed view of the metrics of the event processor, including min, max, mean, median, of the durations and rate metrics for all the counters listed on the /metrics#events page.

The /metrics#events page provides the following metrics about the HMS event processor.

Name	Description
events-processor.avg-events-fetch-duration	Average duration to fetch a batch of events and process it.
events-processor.avg-events-process-duration	Average time taken to process a batch of events received from the Metastore.
events-processor.events-received	Total number of the Metastore events received.
events-processor.events-received-15min-rate	Exponentially weighted moving average (EWMA) of number of events received in last 15 min. This rate of events can be used to determine if there are spikes in event processor activity during certain hours of the day.
events-processor.events-received-1min-rate	Exponentially weighted moving average (EWMA) of number of events received in last 1 min. This rate of events can be used to determine if there are spikes in event processor activity during certain hours of the day.
events-processor.events-received-5min-rate	Exponentially weighted moving average (EWMA) of number of events received in last 5 min. This rate of events can be used to determine if there are spikes in event processor activity during certain hours of the day.
events-processor.events-skipped	Total number of the Metastore events skipped. Events can be skipped based on certain flags are table and database level. You can use this metric to make decisions, such as: • If most of the events are being skipped, see if you might just turn off the event processing. • If most of the events are not skipped, see if you need to add flags on certain databases.
events-processor.status	Metastore event processor status to see if there are events being received or not. Possible states are: • PAUSED The event processor is paused because catalog is being reset concurrently. • ACTIVE The event processor is scheduled at a given frequency. • ERROR • The event processor is in error state and event processing has stopped. • NEEDS_INVALIDATE The event processor could not resolve certain events and needs a manual INVALIDATE command to reset the state. • STOPPED The event processing has been shutdown. No events will be processed. • DISABLED The event processor is not configured to run.

Configuring Event Based Automatic Metadata Sync

As the first step to use the HMS event based metadata sync, enable and configure HMS notifications in Cloudera Manager.

Procedure

1. In Cloudera Manager, navigate to ClustersHive.
2. Navigate to ConfigurationFiltersSCOPEHive Metastore Server.

3. In Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml, click + to expand and enter the following:
   - Name: hive.metastore.notifications.add.thrift.objects
   - Value: true
   - Name: hive.metastore.alter.notifications.basic
   - Value: false
   - Name: hive.metastore.dml.events
   - Value: true
4. Click Save Changes.
5. Navigate to FiltersSCOPEHive-1 (Service-Wide).
6. Select Enable Metastore Notifications for DML Operations.
7. Click Save Changes.
8. If you want the INSERT events are generated when the Spark (and other non-Hive) applications insert data into existing tables and partitions:
   a. Navigate to FiltersSCOPEGateway.
   b. In Hive Client Advanced Configuration Snippet (Safety Valve) for hive-site.xml, click + to expand and enter the following:
      - Name: hive.metastore.dml.events
      - Value: true
   c. Click Save Changes.
9. Restart stale services.

Setting Timeouts in Impala

Depending on how busy your cluster is, you might increase or decrease various timeout values. Increase timeouts if Impala is cancelling operations prematurely, when the system is responding slower than usual but the operations are still successful if given extra time. Decrease timeouts if operations are idle or hanging for long periods, and the idle or hung operations are consuming resources and reducing concurrency.

Setting Timeout and Retries for Thrift Connections to Backend Client

Impala connections to the backend client are subject to failure in cases when the network is momentarily overloaded.

About this task

To avoid failed queries due to transient network problems, you can configure the number of Thrift connection retries using the following option:

Procedure

1. In Cloudera Manager, navigate to Impala serviceConfiguration.

2. In the Impala Daemon Command Line Argument Advanced Configuration Snippet (Safety Valve) field, specify the following.

   To avoid failed queries due to transient network problems, you can configure the number of Thrift connection retries using the following option:

   - The --backend_client_connection_num_retries option specifies the number of times Impala will try connecting to the backend client after the first connection attempt fails. By default, `impalad` will attempt three re-connections before it returns a failure.

   You can configure timeouts for sending and receiving data from the backend client. Therefore, if for some reason a query does not respond, instead of waiting indefinitely for a response, Impala will terminate the connection after a configurable timeout.

   - The --backend_client_rpc_timeout_ms option can be used to specify the number of milliseconds Impala should wait for a response from the backend client before it terminates the connection and signals a failure. The default value for this property is 300000 milliseconds, or 5 minutes.
3. Click Save Changes and restart Impala.

Increasing StateStore Timeout

If you have an extensive Impala schema, for example, with hundreds of databases, tens of thousands of tables, you might encounter timeout errors during startup as the Impala catalog service broadcasts metadata to all the Impala nodes using the StateStore service. To avoid such timeout errors on startup, increase the StateStore timeout value from its default of 10 seconds.

About this task

Increase the timeout value of the StateStore service if you see messages in the impalad log such as:

```
Connection with state-store lost
Trying to re-register with state-store
```

Procedure

1. In Cloudera Manager, navigate to Impala serviceConfiguration.
2. In the search field, type -statestore_subscriber_timeout_seconds.
3. In the StateStoreSubscriber Timeout field, specify a new timeout value larger than the current value.
4. Click Save Changes and restart Impala.

Setting the Idle Query and Idle Session Timeouts

To keep long-running queries or idle sessions from tying up cluster resources, you can set timeout intervals for both individual queries, and entire sessions.

About this task

Procedure

1. In Cloudera Manager, navigate to Impala serviceConfiguration.
2. In the search field, type idle.

3. In the Idle Query Timeout field, specify the time in seconds after which an idle query is cancelled.

   This could be a query whose results were all fetched but was never closed, or one whose results were partially fetched and then the client program stopped requesting further results. This condition is most likely to occur in a client program using the JDBC or ODBC interfaces, rather than in the interactive `impala-shell` interpreter. Once a query is cancelled, the client program cannot retrieve any further results from the query.

   You can reduce the idle query timeout by using the QUERY_TIMEOUT_S query option at the query level. Any non-zero value specified in this field serves as an upper limit for the QUERY_TIMEOUT_S query option.

   The value of 0 disables query timeouts.
4. In the Idle Session Timeout field, specify the time in seconds after which an idle session expires.

   A session is idle when no activity is occurring for any of the queries in that session, and the session has not started any new queries. Once a session is expired, you cannot issue any new query requests to it. The session remains open, but the only operation you can perform is to close it.

   The default value of 0 specifies sessions never expire.

   You can override this setting with the IDLE_SESSION_TIMEOUT query option at the session or query level.
5. Click Save Changes and restart Impala.

Results

Impala checks periodically for idle sessions and queries to cancel. The actual idle time before cancellation might be up to 50% greater than the specified configuration setting. For example, if the timeout setting was 60, the session or query might be cancelled after being idle between 60 and 90 seconds.

Configuring Load Balancer for Impala

For most clusters that have multiple users and production availability requirements, you might want to set up a load-balancing proxy server to relay requests to and from Impala.

When using a load balancer for Impala, applications connect to a single well-known host and port, rather than keeping track of the hosts where a specific Impala daemon is running. The load balancer also lets the Impala nodes share resources to balance out the work loads.

Set up a software package of your choice to perform these functions.

Most considerations for load balancing and high availability apply to the impalad daemons. The statestored and cata logd daemons do not have special requirements for high availability, because problems with those daemons do not result in data loss.

The following are the general setup steps that apply to any load-balancing proxy software:

1. Select and download a load-balancing proxy software or other load-balancing hardware appliance. It should only need to be installed and configured on a single host, typically on an edge node.
2. Configure the load balancer (typically by editing a configuration file). In particular:
   - To relay Impala requests back and forth, set up a port that the load balancer will listen on.
   - Select a load balancing algorithm.
   - For Kerberized clusters, follow the instructions in Special Proxy Considerations for Clusters Using Kerberos on page 61 below.
3. If you are using Hue or JDBC-based applications, you typically set up load balancing for both ports 21000 and 21050, because these client applications connect through port 21050 while the `impala-shell` command connects through port 21000. See Ports used by Impala for when to use port 21000, 21050, or another value depending on what type of connections you are load balancing.
4. Run the load-balancing proxy server, pointing it at the configuration file that you set up.
5. In Cloudera Manager, navigate to ImpalaConfigurationImpala Daemon Default Group.

6. In the Impala Daemons Load Balancer field, specify the address of the load balancer in the *HOST*:*PORT* format.

   This setting lets Cloudera Manager route all appropriate Impala-related operations through the load-balancing proxy server.
7. For any scripts, jobs, or configuration settings for applications that formerly connected to a specific `impalad` to run Impala SQL statements, change the connection information (such as the -i option in `impala-shell`) to point to the load balancer instead.

Note: The following sections use the HAProxy software as a representative example of a load balancer that you can use with Impala. For information specifically about using Impala with the F5 BIG-IP load balancer, see Impala HA with F5 BIG-IP.

Choosing the Load-Balancing Algorithm

Load-balancing software offers a number of algorithms to distribute requests. Each algorithm has its own characteristics that make it suitable in some situations but not others.

Note: Source IP Persistence is required for setting up high availability with Hue.

Leastconn

Connects sessions to the coordinator with the fewest connections, to balance the load evenly. Typically used for workloads consisting of many independent, short-running queries. In configurations with only a few client machines, this setting can avoid having all requests go to only a small set of coordinators.

Recommended for Impala with F5.

Source IP Persistence

Sessions from the same IP address always go to the same coordinator. A good choice for Impala workloads containing a mix of queries and DDL statements, such as CREATE TABLE and ALTE R TABLE. Because the metadata changes from a DDL statement take time to propagate across the cluster, prefer to use the Source IP Persistence algorithm in this case. If you are unable to choose Source IP Persistence, run the DDL and subsequent queries that depend on the results of the DDL through the same session, for example by running impala-shell -f *SCRIPT_FILE* to submit several statements through a single session.

Round-robin

Distributes connections to all coordinator nodes. Typically not recommended for Impala.

You might need to perform benchmarks and load testing to determine which setting is optimal for your use case. Always set up using two load-balancing algorithms: Source IP Persistence for Hue and Leastconn for others.

Special Proxy Considerations for Clusters Using Kerberos

In a cluster using Kerberos, applications check host credentials to verify that the host they are connecting to is the same one that is actually processing the request.

In Impala 2.11 and lower versions, once you enable a proxy server in a Kerberized cluster, users will not be able to connect to individual impala daemons directly from impala-shell.

In Impala 2.12 and higher, when you enable a proxy server in a Kerberized cluster, users have an option to connect to Impala daemons directly from `impala-shell` using the -b / --kerberos_host_fqdn `impala-shell` flag. This option can be used for testing or troubleshooting purposes, but not recommended for live production environments as it defeats the purpose of a load balancer/proxy.

Example:

```
impala-shell -i impalad-1.mydomain.com -k -b loadbalancer-1.mydomain.com
```

Alternatively, with the fully qualified configurations:

```
impala-shell --impalad=impalad-1.mydomain.com:21000 --kerberos --kerberos_ho
st_fqdn=loadbalancer-1.mydomain.com
```

See Impala Shell Configuration Options on page 67 for information about the option.

To validate the load-balancing proxy server, perform these extra Kerberos setup steps:

1. This section assumes you are starting with a Kerberos-enabled cluster. See Configuring Kerberos Authentication on page 24 for instructions for setting up Impala with Kerberos.
2. Choose the host you will use for the proxy server. Based on the Kerberos setup procedure, it should already have an entry impala/*PROXY_HOST@REALM* in its keytab.
3. In Cloudera Manager, navigate to ImpalaConfigurationImpala Daemon Default Group.
4. In the Impala Daemons Load Balancer field, specify the address of the load balancer in the *HOST*:*PORT* format.

   When this field is specified and Kerberos is enabled, Cloudera Manager adds a principal for impa la/*PROXY_HOST@REALM* to the keytab for all Impala daemons.
5. Restart the Impala service.

Client Connection to Proxy Server in Kerberized Clusters

When a client connect to Impala, the service principal specified by the client must match the -principal setting, impa la/*PROXY_HOST@REALM*, of the Impala proxy server as specified in its keytab. And the client should connect to the proxy server port.

In hue.ini, set the following to configure Hue to automatically connect to the proxy server:

```
[impala]
server_host=PROXY_HOST
impala_principal=impala/PROXY_HOST
```

The following are the JDBC connection string formats when connecting through the load balancer with the load balancer's host name in the principal:

```
jdbc:hive2://PROXY_HOST:LOAD_BALANCER_PORT/;principal=impala/_HOST@REALM
jdbc:hive2://PROXY_HOST:LOAD_BALANCER_PORT/;principal=impa
la/PROXY_HOST@REALM
```

When starting `impala-shell`, specify the service principal via the -b or --kerberos_host_fqdn flag.

Special Proxy Considerations for TLS/SSL Enabled Clusters

When TLS/SSL is enabled for Impala, the client application, whether impala-shell, Hue, or something else, expects the certificate common name (CN) to match the hostname that it is connected to. With no load balancing proxy server, the hostname and certificate CN are both that of the impalad instance. However, with a proxy server, the certificate presented by the impalad instance does not match the load balancing proxy server hostname. If you try to load-balance a TLS/SSL-enabled Impala installation without additional configuration, you see a certificate mismatch error when a client attempts to connect to the load balancing proxy host.

You can configure a proxy server in several ways to load balance TLS/SSL enabled Impala:

TLS/SSL Bridging

In this configuration, the proxy server presents a TLS/SSL certificate to the client, decrypts the client request, then re-encrypts the request before sending it to the backend impalad. The client and server certificates can be managed separately. The request or resulting payload is encrypted in transit at all times.

TLS/SSL Passthrough

In this configuration, traffic passes through to the backend impalad instance with no interaction from the load balancing proxy server. Traffic is still encrypted end-to-end.

The same server certificate, utilizing either wildcard or Subject Alternate Name (SAN), must be installed on each impalad instance.

TLS/SSL Offload

In this configuration, all traffic is decrypted on the load balancing proxy server, and traffic between the backend impalad instances is unencrypted. This configuration presumes that cluster hosts reside on a trusted network and only external client-facing communication need to be encrypted in-transit.

If you plan to use Auto-TLS, your load balancer must perform TLS/SSL bridging or TLS/SSL offload.

Refer to your load balancer documentation for the steps to set up Impala and the load balancer using one of the options above.

For information specifically about using Impala with the F5 BIG-IP load balancer with TLS/SSL enabled, see Impala HA with F5 BIG-IP.

Example of Configuring HAProxy Load Balancer for Impala

If you are not already using a load-balancing proxy, you can experiment with HAProxy a free, open source load balancer.

Attention: HAProxy is not a CDH component, and Cloudera does not provide the support for HAProxy. Refer to HAProxy for questions and support issues for HAProxy.

This example shows how you might install and configure that load balancer on a Red Hat Enterprise Linux system.

- Install the load balancer:

```
yum install haproxy
```

- Set up the configuration file: /etc/haproxy/haproxy.cfg. See the following section for a sample configuration file.
- Run the load balancer (on a single host, preferably one not running `impalad`):

```
/usr/sbin/haproxy –f /etc/haproxy/haproxy.cfg
```

- In `impala-shell`, JDBC applications, or ODBC applications, connect to the listener port of the proxy host, rather than port 21000 or 21050 on a host actually running `impalad`. The sample configuration file sets haproxy to listen on port 25003, therefore you would send all requests to *HAPROXY_HOST*:25003.

This is the sample haproxy.cfg used in this example:

```
global
    # To have these messages end up in /var/log/haproxy.log you will
    # need to:
    #
    # 1) configure syslog to accept network log events.  This is done
    #    by adding the '-r' option to the SYSLOGD_OPTIONS in
    #    /etc/sysconfig/syslog
    #
    # 2) configure local2 events to go to the /var/log/haproxy.log
    #   file. A line like the following can be added to
    #   /etc/sysconfig/syslog
    #
    #    local2.*                       /var/log/haproxy.log
    #
    log         127.0.0.1 local0
    log         127.0.0.1 local1 notice
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
```

```
    user        haproxy
    group       haproxy
    daemon

    # turn on stats unix socket
    #stats socket /var/lib/haproxy/stats
#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#
# You might need to adjust timing values to prevent timeouts.
#
# The timeout values should be dependant on how you use the cluster
# and how long your queries run.
#---------------------------------------------------------------------
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    maxconn                 3000
    timeout connect 5000
    timeout client 3600s
    timeout server 3600s

#
# This sets up the admin page for HA Proxy at port 25002.
#
listen stats :25002
    balance
    mode http
    stats enable
    stats auth USERNAME:PASSWORD

# This is the setup for Impala. Impala client connect to load_balancer_hos
t:25003.
# HAProxy will balance connections among the list of servers listed below.
# The list of Impalad is listening at port 21000 for beeswax (impala-shell)
or original ODBC driver.
# For JDBC or ODBC version 2.x driver, use port 21050 instead of 21000.
listen impala :25003
    mode tcp
    option tcplog
    balance leastconn

    server SYMBOLIC_NAME_1 impala-host-1.example.com:21000 check
    server SYMBOLIC_NAME_2 impala-host-2.example.com:21000 check
    server SYMBOLIC_NAME_3 impala-host-3.example.com:21000 check
    server SYMBOLIC_NAME_4 impala-host-4.example.com:21000 check

# Setup for Hue or other JDBC-enabled applications.
# In particular, Hue requires sticky sessions.
# The application connects to load_balancer_host:21051, and HAProxy balances
# connections to the associated hosts, where Impala listens for JDBC
# requests on port 21050.
listen impalajdbc :21051
    mode tcp
    option tcplog
    balance source
    server SYMBOLIC_NAME_5 impala-host-1.example.com:21050 check
```

```
    server SYMBOLIC_NAME_6 impala-host-2.example.com:21050 check
    server SYMBOLIC_NAME_7 impala-host-3.example.com:21050 check
    server SYMBOLIC_NAME_8 impala-host-4.example.com:21050 check
```

Important: Hue requires the check option at the end of each line in the above file to ensure HAProxy can detect any unreachable impalad server, and failover can be successful. Without the TCP check, you can hit an error when the `impalad` daemon to which Hue tries to connect is down.

Note: If your JDBC or ODBC application connects to Impala through a load balancer such as haproxy, be cautious about reusing the connections. If the load balancer has set up connection timeout values, either check the connection frequently so that it never sits idle longer than the load balancer timeout value, or check the connection validity before using it and create a new one if the connection has been closed.

Related Information

Ports used by Impala

Configuring Client Access to Impala

Application developers have a number of options to interface with Impala.

The core development language with Impala is SQL. You can also use Java or other languages to interact with Impala through the standard JDBC and ODBC interfaces used by many business intelligence tools. For specialized kinds of analysis, you can supplement the Impala built-in functions by writing user-defined functions in C++ or Java.

You can connect and submit requests to the Impala through:

- The impala-shell interactive command interpreter
- The Hue web-based user interface
- JDBC
- ODBC
- Impyla

Impala clients can connect to any Coordinator Impala Daemon (impalad) via HiveServer2 over HTTP or over the TCP binary or via Beeswax. All interfaces support Kerberos and LDAP for authentication to Impala. See below for the default ports and the Impala configuration field names to change the ports in Cloudera Manager.

Protocol	Default Port	Cloudera Manager Field to Specify an Alternate Port
HiveServer2 HTTP	28000	Impala Daemon HiveServer2 HTTP Port
HiveServer2 binary TCP	21050	Impala Daemon HiveServer2 Port
Beeswax	21000	Impala Daemon Beeswax Port

Impala Startup Options for Client Connections

Use the following flags to control client connections to Impala when starting Impala Daemon coordinator. If a configuration field exists in Cloudera Manager, the field name is shown in parenthesis next to the flag name.

If the Cloudera Manager interface does not yet have a form field for an option, the Advanced category page for each daemon includes one or more Safety Valve fields where you can enter option names directly.

--accepted_client_cnxn_timeout

Controls how Impala treats new connection requests if it has run out of the number of threads configured by --fe_service_threads.

If --accepted_client_cnxn_timeout > 0, new connection requests are rejected if Impala can't get a server thread within the specified (in seconds) timeout.

If --accepted_client_cnxn_timeout=0, i.e. no timeout, clients wait indefinitely to open the new session until more threads are available.

The default timeout is 5 minutes.

The timeout applies only to client facing thrift servers, i.e., HS2 and Beeswax servers.

--disconnected_session_timeout

When a HiveServer2 session has had no open connections for longer than this value, the session will be closed, and any associated queries will be unregistered.

Specify the value in hours.

The default value is 1 hour.

This flag does not apply to Beeswax clients. When a Beeswax client connection is closed, Impala closes the session associated with that connection.

--fe_service_threads (Impala Daemon Max Client)

Specifies the maximum number of concurrent client connections allowed. The default value is 64 with which 64 queries can run simultaneously.

If you have more clients trying to connect to Impala than the value of this setting, the later arriving clients have to wait for the duration specified by --accepted_client_cnxn_timeout. You can increase this value to allow more client connections. However, a large value means more threads to be maintained even if most of the connections are idle, and it could negatively impact query latency. Client applications should use the connection pool to avoid need for large number of sessions.

--idle_client_poll_time_s

The value of this setting specifies how frequently Impala polls to check if a client connection is idle and closes it if the connection is idle. A client connection is idle if all sessions associated with the client connection are idle.

By default, --idle_client_poll_time_s is set to 30 seconds.

If --idle_client_poll_time_s is set to 0, idle client connections stay open until explicitly closed by the clients.

The connection will only be closed if all the associated sessions are idle or closed. Sessions cannot be idle unless either the flag --idle_session_timeout or the IDLE_SESSION_TIMEOUT query option is set to greater than 0. If idle session timeout is not configured, a session cannot become idle by definition, and therefore its connection stays open until the client explicitly closes it.

--max_cookie_lifetime_s

Starting in Impala 3.4.0, Impala uses cookies for authentication when clients connect via HiveServer2 over HTTP. Use the --max_cookie_lifetime_s startup flag to control how long generated cookies are valid for.

Specify the value in seconds.

The default value is 1 day.

Setting the flag to 0 disables cookie support.

When an unexpired cookie is successfully verified, the user name contained in the cookie is set on the connection.

Each impalad uses its own key to generate the signature, so clients that reconnect to a different impa lad have to re-authenticate.

On a single impalad, cookies are valid across sessions and connections.

--beeswax_port (Impala Daemon Beeswax Port)

Specifies the port for clients to connect to Impala daemon via the Beeswax protocol.

You can disable the Beeswax end point for clients by setting the flag to 0.

--hs2_http_port (Impala Daemon HiveServer2 HTTP Port)

Specifies the port for clients to connect to Impala daemon over HTTP.

You can disable the HTTP end point for clients by setting the flag to 0.

To enable TLS/SSL for HiveServer2 HTTP endpoint, use --ssl_server_certificate and --ssl_private_key.

--hs2_port (Impala Daemon HiveServer2 Port)

Specifies the port for clients to connect to Impala daemon via the HiveServer2 protocol.

You can disable the binary HiveServer2 end point for clients by setting the flag to 0.

Impala Shell Tool

You can use the Impala shell tool (impala-shell) to set up databases and tables, insert data, and issue queries.

For ad-hoc queries and exploration, you can submit SQL statements in an interactive session. To automate your work, you can specify command-line options to process a single statement or a script file. The `impala-shell` accepts all the same SQL statements, plus some shell-only commands that you can use for tuning performance and diagnosing problems.

Cloudera Manager installs `impala-shell` automatically. You might install `impala-shell` manually on other systems not managed by Cloudera Manager, so that you can issue queries from client systems that are not also running the Impala daemon or other Apache Hadoop components.

Impala Shell Configuration Options

You can specify the following options when starting impala-shell to control how shell commands are executed. You can specify options on the command line or in the impala-shell configuration file.

Command-Line Option	Configuration File Setting	Explanation
-B or --delimited	write_delimited=true	Causes all query results to be printed in plain format as a delimited text file. Useful for producing data files to be used with other Hadoop components. Also useful for avoiding the performance overhead of pretty-printing all output, especially when running benchmark tests using queries returning large result sets. Specify the delimiter character with the --output_delimiter option. Store all query results in a file rather than printing to the screen with the -B option.
--live_progress	live_progress=true	Prints a progress bar showing roughly the percentage complete for each query. Information is updated interactively as the query progresses.
--disable_live_progress	live_progress=false	Disables live_progress in the interactive mode.
-b or --kerberos_host_fqdn	kerberos_host_fqdn= *LOAD-BALANCER-HOSTNAME*	If set, the setting overrides the expected hostname of the Impala daemon's Kerberos service principal. `impala-shell` will check that the server's principal matches this hostname. This may be used when impalad is configured to be accessed via a load-balancer, but it is desired for impala-shell to talk to a specific impalad directly.
--print_header	print_header=true	
-o *FILENAME* or --output_file *FILENAME*	output_file=*FILENAME*	Stores all query results in the specified file. Typically used to store the results of a single query issued from the command line with the -q option. Also works for interactive sessions; you see the messages such as number of rows fetched, but not the actual result set. To suppress these incidental messages when combining the -q and -o options, redirect stderr to /dev/null.

Command-Line Option	Configuration File Setting	Explanation
--output_delimiter= *CHARACTER*	output_delimiter=*CHARACTER*	Specifies the character to use as a delimiter between fields when query results are printed in plain format by the -B option. Defaults to tab ('\t'). If an output value contains the delimiter character, that field is quoted, escaped by doubling quotation marks, or both.
-p or --show_profiles	show_profiles=true	Displays the query execution plan (same output as the EXPLAIN statement) and a more detailed low-level breakdown of execution steps, for every query executed by the shell.
-h or --help	N/A	Displays help information.
N/A	history_max=1000	Sets the maximum number of queries to store in the history file.
-i *HOSTNAME* or -- impalad=*HOSTNAME*[:*PORTNUM*]	impalad=*HOSTNAME*[:*PORTNUM*]	Connects to the impalad daemon on the specified host. The default port of 21000 is assumed unless you provide another value. You can connect to any host in your cluster that is running impalad. If you connect to an instance of impalad that was started with an alternate port specified by the --fe_port flag, provide that alternative port.
-q *QUERY* or --query=*QUERY*	query=*QUERY*	Passes a query or other `impala-shell` command from the command line. The `impala-shell` interpreter immediately exits after processing the statement. It is limited to a single statement, which could be a SELECT, CREATE TABLE, SHOW TABLES, or any other statement recognized in impala-shell. Because you cannot pass a USE statement and another query, fully qualify the names for any tables outside the default database. (Or use the -f option to pass a file with a USE statement followed by other queries.)
-f *QUERY_FILE* or --query_file=*QUERY_FILE*	query_file=*PATH_TO_QUERY_FILE*	Passes a SQL query from a file. Multiple statements must be semicolon (;) delimited.
-k or --kerberos	use_kerberos=true	Kerberos authentication is used when the shell connects to impalad. If Kerberos is not enabled on the instance of impalad to which you are connecting, errors are displayed.
--query_option= "*OPTION=VALUE* -Q "*OPTION=VALUE*"	Header line [impala.query_options], followed on subsequent lines by *OPTION=VALUE*, one option per line.	Sets default query options for an invocation of the `impala-shell` command. To set multiple query options at once, use more than one instance of this command-line option. The query option names are not case-sensitive.
-s *KERBEROS_SERVICE_NAME* or --kerberos_service_name= *NAME*	kerberos_service_name=*NAME*	Instructs impala-shell to authenticate to a particular impalad service principal. If a *KERBEROS_SERVICE_NAME* is not specified, impala is used by default. If this option is used in conjunction with a connection in which Kerberos is not supported, errors are returned.
-V or --verbose	verbose=true	Enables verbose output.
--quiet	verbose=false	Disables verbose output.
-v or --version	version=true	Displays version information.
-c	ignore_query_failure=true	Continues on query failure.
-d *DEFAULT_DB* or --database=*DEFAULT_DB*	default_db=*DEFAULT_DB*	Specifies the database to be used on startup. Same as running the USE statement after connecting. If not specified, a database named DEFAULT is used.
--ssl	ssl=true	Enables TLS/SSL for `impala-shell`.

Command-Line Option	Configuration File Setting	Explanation
PATH_TO_CERTIFICATE	ca_cert=*PATH_TO_CERTIFICATE*	The local pathname pointing to the third-party CA certificate, or to a copy of the server certificate for self-signed server certificates. If --ca_cert is not set, `impala-shell` enables TLS/SSL, but does not validate the server certificate. This is useful for connecting to a known-good Impala that is only running over TLS/SSL, when a copy of the certificate is not available (such as when debugging customer installations).
-l	use_ldap=true	Enables LDAP authentication.
-u	user=*USER_NAME*	Supplies the username, when LDAP authentication is enabled by the -l option. (Specify the short username, not the full LDAP distinguished name.) The shell then prompts interactively for the password.
--ldap_password_cmd= *COMMAND*	N/A	Specifies a command to run to retrieve the LDAP password, when LDAP authentication is enabled by the -l option. If the command includes space-separated arguments, enclose the command and its arguments in quotation marks.
--config_file= *PATH_TO_CONFIG_FILE*	N/A	Specifies the path of the file containing `impala-shell` configuration settings. The default is /etc/impalarc. This setting can only be specified on the command line.
--live_progress	N/A	Prints a progress bar showing roughly the percentage complete for each query. The information is updated interactively as the query progresses.
--live_summary	N/A	Prints a detailed report, similar to the SUMMARY command, showing progress details for each phase of query execution. The information is updated interactively as the query progresses.
--var= *VARIABLE_NAME*= *VALUE*	N/A	Defines a substitution variable that can be used within the `impala-shell` session. The variable can be substituted into statements processed by the -q or -f options, or in an interactive shell session. Within a SQL statement, you substitute the value by using the notation ${var:*VARIABLE_NAME*}.
--auth_creds_ok_in_clear	N/A	Allows LDAP authentication to be used with an insecure connection to the shell. WARNING: This will allow authentication credentials to be sent unencrypted, and hence may be vulnerable to an attack.
--protocol= *PROTOCOL*	N/A	Protocol to use for the connection to Impala. Valid *PROTOCOL* values are: • 'hs2': Impala-shell uses the binary TCP based transport to speak to the Impala Daemon via the HiveServer2 protocol. • 'hs2-http': Impala-shell uses HTTP transport to speak to the Impala Daemon via the HiveServer2 protocol. • 'beeswax': Impala-shell uses the binary TCP based transport to speak to the Impala Daemon via Beeswax. This is the current default setting. You cannot connect to the 3.2 or earlier versions of Impala using the 'hs2' or 'hs2-http' option. Beeswax support is deprecated and will be removed in the future.

Impala Shell Configuration File

You can store a set of default settings for `impala-shell` in the `impala-shell` configuration file.

The global `impala-shell` configuration file is located in /etc/impalarc.

The user-level `impala-shell` configuration file is located in ~/.impalarc.

Note that the global-level file name is different from the user-level file name. The global-level file name does not include a dot (.) in the file name.

The default path of the global configuration file can be changed by setting the $IMPALA_SHELL_GLOBAL_CONFIG_FILE environment variable.

To specify a different file name or path for the user-level configuration file, start `impala-shell` with the --config_file `impala-shell` option set to the path of the configuration file.

Typically, an administrator creates the global configuration file for the `impala-shell`, and if the user-level configuration file exists, the options set in the user configuration file take precedence over those in the global configuration file.

In turn, any options you specify on the `impala-shell` command line override any corresponding options within the configuration file.

The `impala-shell` configuration file (global or user) must contain a header label [impala], followed by the options specific to `impala-shell`.

The `impala-shell` configuration file consists of key-value pairs, one option per line. Everything after the # character on a line is treated as a comment and ignored.

The names of the options in the configuration file are similar (although not necessarily identical) to the long-form command-line arguments to the `impala-shell` command. For the supported options in the configuration file, see Impala Shell Configuration Options on page 67.

You can specify key-value pair options using keyval, similar to the --var command-line option. For example, keyval=*VARIABLE1*=*VALUE1*.

The query options specified in the [impala] section override the options specified in the [impala.query_options] section.

The following example shows a configuration file that you might use during benchmarking tests. It sets verbose mode, so that the output from each SQL query is followed by timing information. `impala-shell` starts inside the database containing the tables with the benchmark data, avoiding the need to issue a USE statement or use fully qualified table names.

In this example, the query output is formatted as delimited text rather than enclosed in ASCII art boxes, and is stored in a file rather than printed to the screen. Those options are appropriate for benchmark situations, so that the overhead of `impala-shell` formatting and printing the result set does not factor into the timing measurements. It also enables the show_profiles option. That option prints detailed performance information after each query, which might be valuable in understanding the performance of benchmark queries.

```
[impala]
verbose=true
default_db=tpc_benchmarking
write_delimited=true
output_delimiter=,
output_file=/home/tester1/benchmark_results.csv
show_profiles=true
keyval=msg1=hello,keyval=msg2=world
```

The following example shows a configuration file that connects to a specific remote Impala node, runs a single query within a particular database, then exits. Any query options predefined under the [impala.query_options] section in the configuration file take effect during the session.

You would typically use this kind of single-purpose configuration setting with the `impala-shell` command-line option --config_file=*PATH_TO_CONFIG_FILE*, to easily select between many predefined queries that could be run against different databases, hosts, or even different clusters. To run a sequence of statements instead of a single query, specify the configuration option query_file=*PATH_TO_QUERY_FILE* instead.

```
[impala]
impalad=impala-test-node1.example.com
default_db=site_stats
# Issue a predefined query and immediately exit.
query=select count(*) from web_traffic where event_date = trunc(now(),'dd')

[impala.query_options]
```

```
mem_limit=32g
```

Connecting to Impala Daemon in Impala Shell

In an `impala-shell` session, you need to connect to an `impalad` daemon to issue queries. When you connect to an impalad, and that daemon coordinates the execution of all queries sent to it.

About this task

Specify the connection information using the following options:

- Through command-line options when you run the `impala-shell` command.
- Through a configuration file that is read when you run the `impala-shell` command.
- During an `impala-shell` session, by issuing a CONNECT command.

Note: You cannot connect to the 3.2 or earlier versions of Impala using the 'hs2' or 'hs2-http' protocol (--protocol option).

To connect the Impala shell during shell startup:

1. Locate the hostname that is running an instance of the `impalad` daemon. If that `impalad` uses a non-default port (something other than port 21000) for `impala-shell` connections, find out the port number also.
2. Use the -i option to the `impala-shell` interpreter to specify the connection information for that instance of `impalad`:

```
# When you are connecting to an impalad running on the same machine.
# The prompt will reflect the current hostname.
$ impala-shell
# When you are connecting to an impalad running on a remote machine, and
 impalad is listening
# on a non-default port over the HTTP HiveServer2 protocol.
$ impala-shell -i SOME.OTHER.HOSTNAME:PORT_NUMBER --protocol='hs2-http'
# When you are connecting to an impalad running on a remote machine, and
impalad is listening
# on a non-default port.
$ impala-shell -i SOME.OTHER.HOSTNAME:PORT_NUMBER
```

To connect to an Impala in the`impala-shell` session:

1. Start the Impala shell with no connection:

```
impala-shell
```

2. Locate the hostname that is running the `impalad` daemon. If that `impalad` uses a non-default port (something other than port 21000) for `impala-shell` connections, find out the port number also.
3. Use the connect command to connect to an Impala instance. Enter a command and replace *IMPALAD-HOST* with the hostname you have configured to run Impala in your environment.

```
[Not connected] > connect IMPALAD-HOST
[IMPALAD-HOST:21000] >
```

To start `impala-shell` in a specific database:

You can use all the same connection options as in previous examples. For simplicity, these examples assume that you are logged into one of the Impala daemons.

1. Find the name of the database containing the relevant tables, views, and so on that you want to operate on.
2. Use the -d option to the `impala-shell` interpreter to connect and immediately switch to the specified database, without the need for a USE statement or fully qualified names:

```
# Subsequent queries with unqualified names operate on
# tables, views, and so on inside the database named 'staging'.
```

```
$ impala-shell -i localhost -d staging

# It is common during development, ETL, benchmarking, and so on
# to have different databases containing the same table names
# but with different contents or layouts.
$ impala-shell -i localhost -d parquet_snappy_compression
$ impala-shell -i localhost -d parquet_gzip_compression
```

To run one or several statements in non-interactive mode:

You can use all the same connection options as in previous examples. For simplicity, these examples assume that you are logged into one of the Impala daemons.

1. Construct a statement, or a file containing a sequence of statements, that you want to run in an automated way, without typing or copying and pasting each time.
2. Invoke `impala-shell` with the -q option to run a single statement, or the -f option to run a sequence of statements from a file. The `impala-shell` command returns immediately, without going into the interactive interpreter.

```
# A utility command that you might run while developing shell scripts
# to manipulate HDFS files.
$ impala-shell -i localhost -d database_of_interest -q 'show tables'

# A sequence of CREATE TABLE, CREATE VIEW, and similar DDL statements
# can go into a file to make the setup process repeatable.
$ impala-shell -i localhost -d database_of_interest -f recreate_tables.sql
```

Running Commands and SQL Statements in Impala Shell

This topic provides the commonly used syntax and shortcut keys in impala-shell.

The following are a few of the key syntax and usage rules for running commands and SQL statements in impala-shell.

- To see the full set of available commands, press TAB twice.
- To cycle through and edit previous commands, click the up-arrow and down-arrow keys.
- Use the standard set of keyboard shortcuts in GNU Readline library for editing and cursor movement, such as Ctrl-A for the beginning of line and Ctrl-E for the end of line.
- Commands and SQL statements must be terminated by a semi-colon.
- Commands and SQL statements can span multiple lines.
- Use -- to denote a single-line comment and /* */ to denote a multi-line comment.

  A comment is considered part of the statement it precedes, so when you enter a -- or /* */ comment, you get a continuation prompt until you finish entering a statement ending with a semicolon. For example:

```
[impala] > -- This is a test comment
                  > SHOW TABLES LIKE 't*';
```

- If a comment contains the ${*VARIABLE_NAME*} and it is not for a variable substitution, the $ character must be escaped, e.g. -- \${hello}.

Variable Substitution in impala-shell

You can define substitution variables to be used within SQL statements processed by `impala-shell`.

1. You specify the variable and its value as below.
   - On the command line, you specify the option --var=*VARIABLE_NAME*=*VALUE*
   - Within an interactive session or a script file processed by the -f option, use the SET VAR:*VARIABLE_NAME*=*VALUE* command.
2. Use the above variable in SQL statements in the impala-shell session using the notation: ${VA R:*VARIABLE_NAME*}.

For example, here are some `impala-shell` commands that define substitution variables and then use them in SQL statements executed through the -q and -f options. Notice how the -q argument strings are single-quoted to prevent shell expansion of the ${var:value} notation, and any string literals within the queries are enclosed by double quotation marks.

```
$ impala-shell --var=tname=table1 --var=colname=x --var=coltype=string -q '
CREATE TABLE ${var:tname} (${var:colname} ${var:coltype}) STORED AS PARQUET'
Query: CREATE TABLE table1 (x STRING) STORED AS PARQUET
```

The below example shows a substitution variable passed in by the --var option, and then referenced by statements issued interactively. Then the variable is reset with the SET command.

```
$ impala-shell --quiet --var=tname=table1

[impala] > SELECT COUNT(*) FROM ${var:tname};

[impala] > SET VAR:tname=table2;
[impala] > SELECT COUNT(*) FROM ${var:tname};
```

When you run a query, the live progress bar appears in the output of a query. The bar shows roughly the percentage of completed processing. When the query finishes, the live progress bar disappears from the console output.

Impala Shell Command Reference

Use the following commands within impala-shell to pass requests to the impalad daemon that the shell is connected to. You can enter a command interactively at the prompt or pass it as the argument to the -q option of impala-shell.

Command	Explanation
Impala SQL statements	You can issue valid SQL statements to be executed.
connect	Connects to the specified instance of impalad. The default port of 21000 is assumed unless you provide another value. You can connect to any host in your cluster that is running impalad. If you connect to an instance of impalad that was started with an alternate port specified by the --fe_port flag, you must provide that alternate port.
help	Help provides a list of all available commands and options.
history	Maintains an enumerated cross-session command history. This history is stored in the ~/.impalahistory file.
profile	Displays low-level information about the most recent query. Used for performance diagnosis and tuning. The report starts with the same information as produced by the EXPLAIN statement and the SUMMARY command.
quit	Exits the shell. Remember to include the final semicolon so that the shell recognizes the end of the command.
rerun or @	Executes a previous `impala-shell` command again, from the list of commands displayed by the history command. These could be SQL statements, or commands specific to `impala-shell` such as quit or profile. Specify an integer argument. A positive integer N represents the command labelled N in the output of the HIST ORY command. A negative integer -N represents the Nth command from the end of the list, such as -1 for the most recent command. Commands that are executed again do not produce new entries in the HISTORY output list.
set	Manages query options for an `impala-shell` session. These options are used for query tuning and troubleshooting. Issue SET with no arguments to see the current query options, either based on the `impalad` defaults, as specified by you at `impalad` startup, or based on earlier SET statements in the same session. To modify option values, issue commands with the syntax set *OPTION=VALUE*. To restore an option to its default, use the unset command.
shell	Executes the specified command in the operating system shell without exiting impala-shell. You can use the ! character as shorthand for the shell command. **Note:** Quote any instances of the -- or /* tokens to avoid them being interpreted as the start of a comment. To embed comments within source or ! commands, use the shell comment character # before the comment portion of the line.

Command	Explanation
source or src	Executes one or more statements residing in a specified file from the local filesystem. Allows you to perform the same kinds of batch operations as with the -f option, but interactively within the interpreter. The file can contain SQL statements and other `impala-shell` commands, including additional SOURCE commands to perform a flexible sequence of actions. Each command or statement, except the last one in the file, must end with a semicolon.
summary	Summarizes the work performed in various stages of a query. It provides a higher-level view of the information displayed by the EXPLAIN command. Added in Impala 1.4.0. The time, memory usage, and so on reported by SUMMARY only include the portions of the statement that read data, not when data is written. Therefore, the PROFILE command is better for checking the performance and scalability of INSERT statements. You can see a continuously updated report of the summary information while a query is in progress.
unset	Removes any user-specified value for a query option and returns the option to its default value. You can also use it to remove user-specified substitution variables using the notation UNSET VA R:*VARIABLE_NAME*.
use	Indicates the database against which to execute subsequent commands. Lets you avoid using fully qualified names when referring to tables in databases other than default. Not effective with the -q option, because that option only allows a single statement in the argument.
version	Returns Impala version information.

Configuring ODBC for Impala

Download and configure the ODBC driver to integrate your applications with Impala.

About this task

Impala has been tested with the Impala ODBC driver version 2.5.42, and Cloudera recommends that you use this version with the current version of Impala.

Procedure

1. Download and install an ODBC driver.
2. Configure the ODBC port.

   Versions 2.5 and 2.0 of the Cloudera ODBC Connector use the HiveServer2 protocol, corresponding to Impala port 21050.

   Version 1.x of the Cloudera ODBC Connector uses the original HiveServer1 protocol, corresponding to Impala port 21000.

Configuring JDBC for Impala

Download and configure the JDBC driver to access Impala from a Java program that you write, or a Business Intelligence or similar tool that uses JDBC to communicate with database products.

About this task

The following are the default ports that Impala server accepts JDBC connections through:

Protocol	Default Port	Flag to Specify an Alternate Port
HTTP	28000	##hs2_http_port
Binary TCP	21050	##hs2_port

Make sure the port for the protocol you are using is available for communication with clients, for example, that it is not blocked by firewall software.

If your JDBC client software connects to a different port, specify that alternative port number with the flag in the above table when starting the impalad.

Procedure

1. **Configure the JDBC port.**

   Impala server accepts JDBC connections through port 21050 by default. Make sure this port is available for communication with other hosts on your network, for example, that it is not blocked by firewall software. If your JDBC client software connects to a different port, specify that alternative port number in the Impala Daemon HiveServer2 Portfield in Cloudera Manager, in the Configuration tab.
2. Install the JDBC driver.

   Impala has been tested using the Impala JDBC driver version 2.5.45 and 2.6.2. Cloudera recommends that you use one of these two versions with Impala.
3. Enable Impala JDBC support on client systems.
4. Establish JDBC connections.

   The JDBC driver class depends on which driver you select.

   Using the Cloudera JDBC Connector (recommended):

   Depending on the level of the JDBC API your application is targeting, you can use the following fully-qualified class names (FQCNs):

   - com.cloudera.impala.jdbc41.Driver
   - com.cloudera.impala.jdbc41.DataSource

   - com.cloudera.impala.jdbc4.Driver
   - com.cloudera.impala.jdbc4.DataSource

   - com.cloudera.impala.jdbc3.Driver
   - com.cloudera.impala.jdbc3.DataSource

   The connection string has the following format:

   ```
   jdbc:impala://HOST:PORT[/SCHEMA];PROPERTY1=VALUE;PROPERTY2=VALUE;...
   ```

   The port value is typically 21050 for Impala.

   To connect to an instance of Impala that requires Kerberos authentication, use a connection string of the form jdbc:impala://*HOST*:*PORT*/;principal=*PRINCIPAL_NAME*. The principal must be the same user principal you used when starting Impala.

   To connect to an instance of Impala that requires LDAP authentication, use a connection string of the form jdbc:impala://*HOST*:*PORT*/*DB_NAME*;user=*LDAP_USERID*;password=*LDAP_PASSWORD*.

   To connect to an instance of Impala over HTTP, specify the HTTP port, 28000 by default, and transportMode=http in the connection string.

   Note: To establish a connection with an Impala instance from your client, using any authentication mode, you must use the connection string jdbc:impala. Using the connection string jdbc:hive2 is not recommended and is not supported.

   For updated information on the version of the JDBC driver you are using and for connection string examples for different supported authentications, refer to the link provided under Related Information.

Configuring Impyla for Impala

Explains how to install Impyla to connect to and submit SQL queries to Impala. Impyla is a Python client wrapper around the HiveServer2 Thrift Service. It connects to Impala and implements Python DB API 2.0.

About this task

Impyla releases are available at pypi.org. To get the available releases, check Release history.

Note: Cloudera will not support versions of Impyla that are built manually from source code.

Key Features of Impyla

- HiveServer2 compliant.
- Works with Impala including nested data.
- DB API 2.0 (PEP 249)-compliant Python client (similar to sqlite or MySQL clients) supporting Python 2.6+ and Python 3.3+.
- Works with Kerberos, LDAP, SSL.
- SQLAlchemy connector.
- Converts to pandas DataFrame, allowing easy integration into the Python data stack (including scikit-learn and matplotlib); see the Ibis project for a richer experience.
- For more information, see here.

Before you begin

Different systems require different packages to be installed to enable SASL support in Impyla. The following list shows some examples of how to install the packages on different distributions.

You must have the following installed in your environment before installing impyla. Python 2.6+ or 3.3+ and the pip packages six, bitarray, thrift and thriftpy2 will be automatically installed as dependencies when installing impyla. However if you clone the impyla repo and run their local copy, you must install these pip packages manually.

- Install the latest pip and setuptools:

  ```
  python -m pip install --upgrade pip setuptools
  ```

  Optionally, to install Impyla with Hive and/or GSSAPI (kerberos) support, you will also need to install additional software packages:
- RHEL/CentOS:

  ```
  sudo yum install gcc-c++ cyrus-sasl-md5 cyrus-sasl-plain cyrus-sasl-gssapi
   cyrus-sasl-devel
  ```
- Ubuntu:

  ```
  sudo apt install g++ libsasl2-dev libsasl2-2 libsasl2-modules-gssapi-mit
  ```

Procedure

1. Using pip you can install the latest release: pip install impyla
2. Optionally, to install Impyla with GSSAPI (kerberos) support:pip install impyla[kerberos]
3. You also need to pip-install pandas for conversion to DataFrame objects or sqlalchemy for the SQLAlchemy engine.

Example

Sample codes

Impyla implements the Python DB API v2.0 (PEP 249) database interface (refer to it for API details):

```
from impala.dbapi import connect

conn = connect(host = "my.host.com", port = 21050)
cursor = conn.cursor()
cursor.execute("SELECT * FROM mytable LIMIT 100")
print(cursor.description)  # prints the result set's schema
results = cursor.fetchall()
cursor.close()
conn.close()
```

The Cursorobject also exposes the iterator interface, which is buffered (controlled by cursor.arraysize):

```
cursor.execute("SELECT * FROM mytable LIMIT 100")
for row in cursor:
    print(row)
```

Furthermore the Cursor object returns you information about the columns returned in the query. This is useful to export your data as a csv file.

```
import csv

cursor.execute("SELECT * FROM mytable LIMIT 100")
columns = [datum[0] for datum in cursor.description]
targetfile = "/tmp/foo.csv"

with open(targetfile, "w", newline = "") as outcsv:
    writer = csv.writer(
        outcsv,
        delimiter = ",",
        quotechar = '"',
        quoting = csv.QUOTE_ALL,
        lineterminator = "\n")
writer.writerow(columns)
for row in cursor:
    writer.writerow(row)
```

You can also get back a pandas DataFrame object

```
from impala.util import as_pandas

# carry df through scikit-learn, for example
df = as_pandas(cur)
```

Connecting over HTTP/HTTPS

GSSAPI (kerberos) authentication over HTTP has been supported since 0.17a1. Use this example to establish connection over HTTP/HTTPS.

```
from impala.dbapi import connect

conn = connect(
    "impala-coordinator.example.com",
    28000,
    auth_mechanism = "GSSAPI",
    kerberos_service_name = "impala",
    use_http_transport = True,
    http_path = "cliservice",
    auth_cookie_name = "impala.auth")
cursor = conn.cursor()
cursor.execute("SHOW DATABASES")
```

```
res = cursor.fetchall()
cursor.close()
conn.close()
```

Related Information

Running Queries on Impala Tables

Connecting to DataHub

Lists an example code to connect to Impala with LDAP over http using LDAP as the authentication mechanism.

Example

Sample code

```
from impala.dbapi import connect

conn = connect(
    host = "aaaaaaa-aaaa-master0.se-sandb.a465-9q4k.cloudera.site",
    port = 443,
    auth_mechanism = "LDAP",
    use_ssl = True,
    use_http_transport = True,
    http_path = "aaaaaaa-aaaa/cdp-proxy-api/impala",
    user = "aaaaaaa",
    password = "xxxxx")
cursor = conn.cursor()
cursor.execute("SELECT * FROM default.emax_temp")

for row in cursor:
    print(row)
cursor.close()
conn.close()
```

Connecting to DataHub Data Mart

Lists an example code to connect to DataHub Data Mart using LDAP as the authentication mechanism.

Before you begin

Must have the latest Impyla release.

Example

Sample code

```
from impala.dbapi import connect

conn = connect(
    "xxxxxxx-data-mart-master0.xxxxxxx.xcu2-8y8x.dev.cldr.work",
    443,
    auth_mechanism = "LDAP",
    user = "XXXXX",
    password = "XXXXX",
    use_ssl = True,
    use_http_transport = True,
    http_path = "xxxxxxx-data-mart/cdp-proxy-api/impala")
cursor = conn.cursor()
cursor.execute("SHOW DATABASES")
res = cursor.fetchall()
print(res)
cursor.close()
conn.close()
```

Configuring Delegation for Clients

Impala supports user and group delegation for client connections.

About this task

When users submit Impala queries through a separate application, such as Hue or a business intelligence tool, typically all requests are treated as coming from the same user. Impala supports "delegation" where users whose names you specify can delegate the execution of a query to another user. The query runs with the privileges of the delegated user, not the original authenticated user.

You also have an option to delegate using groups. Instead of listing a large number of delegated users, you can create a group of those users and specify the delegated group name in the Impalad startup option. The client sends the delegated user name, and Impala performs an authorization to see if the delegated user belongs to a delegated group.

The name of the delegated user is passed using the HiveServer2 protocol configuration property impala.doas.user when the client connects to Impala.

When the client connects over HTTP, the doAs parameter can be specified in the HTTP path. For example:

```
/?doAs=DELEGATED_USER
```

Currently, the delegation feature is available only for Impala queries submitted through application interfaces such as Hue and BI tools. For example, Impala cannot issue queries using the privileges of the HDFS user.

Attention:

- When the delegation is enabled in Impala, the Impala clients should take an extra caution to prevent unauthorized access for the delegate-able users.
- Impala requires Apache Ranger on the cluster to enable delegation. Without Ranger installed, the delegation feature will fail with the following error: User user1 is not authorized to delegate to user2. User/group delegation is disabled.

Procedure

To enable delegation:

1. In Cloudera Manager, navigate to ClustersImpala.
2. In the Configuration tab, click Impala-1 (Service-Wide) in the Scope and click Security in the Category.
3. In the Proxy User Configuration field, type the a semicolon-separated list of key=value pairs of authorized proxy users to the user(s) they can impersonate.

   The list of delegated users are delimited with a comma, e.g. hue=user1, user2.
4. In the Proxy Group Configuration field, type the a semicolon-separated list of key=value pairs of authorized proxy users to the group(s) they can impersonate.

   The list of delegated groups are delimited with a comma, e.g. hue=group1, group2.
5. Click Save Changes and restart the Impala service.

Spooling Impala Query Results

In Impala, you can control how query results are materialized and returned to clients, e.g. impala-shell, Hue, JDBC apps.

Result spooling is turned off by default, but can be enabled via the SPOOL_QUERY_RESULTS query option.

- When query result spooling is disabled, Impala relies on clients to fetch results to trigger the generation of more result row batches until all the result rows have been produced. If a client issues a query without fetching all the results then that query continues to be in the running state indefinitely and the query fragments continue to consume the resources until the query is cancelled and unregistered, potentially tying up resources and causing other queries to wait for an extended period of time in admission control.

Impala would materialize rows on-demand where rows are created only when the client requests them.

For example, if a Hue user runs a complex query that returns 1000 rows, but does not scroll through all the returned rows, and then stays idle for a while, the query will remain running and will hold onto all of its resources until it is explicitly closed or the session times out.

- When query result spooling is enabled, result sets of queries are eagerly fetched and spooled in the spooling location, either in memory or on disk.

  Once all result rows have been fetched and stored in the spooling location, the resources are freed up. Incoming client fetches can get the data from the spooled results.

Admission Control and Result Spooling

Query results spooling collects and stores query results in memory that is controlled by admission control. Use the following query options to calibrate how much memory to use and when to spill to disk.

MAX_RESULT_SPOOLING_MEM

The maximum amount of memory used when spooling query results. If this value is exceeded when spooling results, all memory will most likely be spilled to disk. Set to 100 MB by default.

MAX_SPILLED_RESULT_SPOOLING_MEM

The maximum amount of memory that can be spilled to disk when spooling query results. Must be greater than or equal to MAX_RESULT_SPOOLING_MEM. If this value is exceeded, the coordinator fragment will block until the client has consumed enough rows to free up more memory. Set to 1 GB by default.

Fetch Timeout

Resources for a query are released when the query completes its execution. To prevent clients from indefinitely waiting for query results, use the FETCH_ROWS_TIMEOUT_MS query option to set the timeout when clients fetch rows. Timeout applies both when query result spooling is enabled and disabled:

- When result spooling is disabled (SPOOL_QUERY_RESULTS = FALSE), the timeout controls how long a client waits for a single row batch to be produced by the coordinator.
- When result spooling is enabled ((SPOOL_QUERY_RESULTS = TRUE), a client can fetch multiple row batches at a time, so this timeout controls the total time a client waits for row batches to be produced.

Explain Plans

Below is the part of the EXPLAIN plan output for result spooling.

```
F01:PLAN FRAGMENT [UNPARTITIONED] hosts=1 instances=1
|  Per-Host Resources: mem-estimate=4.02MB mem-reservation=4.00MB thread-r
eservation=1
PLAN-ROOT SINK
|  mem-estimate=4.00MB mem-reservation=4.00MB spill-buffer=2.00MB thread-res
ervation=0
```

- The mem-estimate for the PLAN-ROOT SINK is an estimate of the amount of memory needed to spool all the rows returned by the query.
- The mem-reservation is the number and size of the buffers necessary to spool the query results. By default, the read and write buffers are 2 MB in size each, which is why the default is 4 MB.

PlanRootSink

In Impala, the PlanRootSink class controls the passing of batches of rows to the clients and acts as a queue of rows to be sent to clients.

- When result spooling is disabled, a single batch or rows is sent to the PlanRootSink, and then the client must consume that batch before another one can be sent.
- When result spooling is enabled, multiple batches of rows can be sent to the PlanRootSink, and multiple batches can be consumed by the client.

Related Information

Impala query options