Cloudera Runtime 7.1.7

Managing Apache Impala

Date published: 2019-11-01
Date modified: 2021-07-12

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

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

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

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

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

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

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

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

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

Contents

Managing Resources in Impala.....4
Admission Control and Query Queuing..... 4
Enabling Admission Control..... 8
Creating Static Pools..... 8
Configuring Dynamic Resource Pool..... 8
Dynamic Resource Pool Settings..... 9
Admission Control Sample Scenario..... 11
Cancelling a Query..... 13

Managing Metadata in Impala.....13
On-demand Metadata..... 13
Automatic Invalidation of Metadata Cache..... 14
Automatic Invalidation/Refresh of Metadata..... 15
Configuring Event Based Automatic Metadata Sync..... 17

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 ClustersImpala Admission Control 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 run 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 run 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 run 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
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_LIMIT_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):

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 INVALIDATE 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 catalogd, Impala invalidates 10% of the least recently used tables.

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

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.