Cloudera Runtime 1.5.5

Apache Impala

Date published: 2020-11-30
Date modified: 2025-06-06

https://docs.cloudera.com/

Legal Notice

Contents

Apache Impala Overview

The Apache Impala provides high-performance, low-latency SQL queries on data stored in popular Apache Hadoop file formats.

The Impala solution is composed of the following components.

Impala

The Impala service coordinates and executes queries received from clients. Queries are distributed among Impala nodes, and these nodes then act as workers, executing parallel query fragments.

Hive Metastore

Stores information about the data available to Impala. For example, the metastore lets Impala know what databases are available and what the structure of those databases is. As you create, drop, and alter schema objects, load data into tables, and so on through Impala SQL statements, the relevant metadata changes are automatically broadcast to all Impala nodes by the dedicated catalog service.

Clients

Entities including Hue, ODBC clients, JDBC clients, Business Intelligence applications, and the Impala Shell can all interact with Impala. These interfaces are typically used to issue queries or complete administrative tasks such as connecting to Impala.

Storage for data to be queried

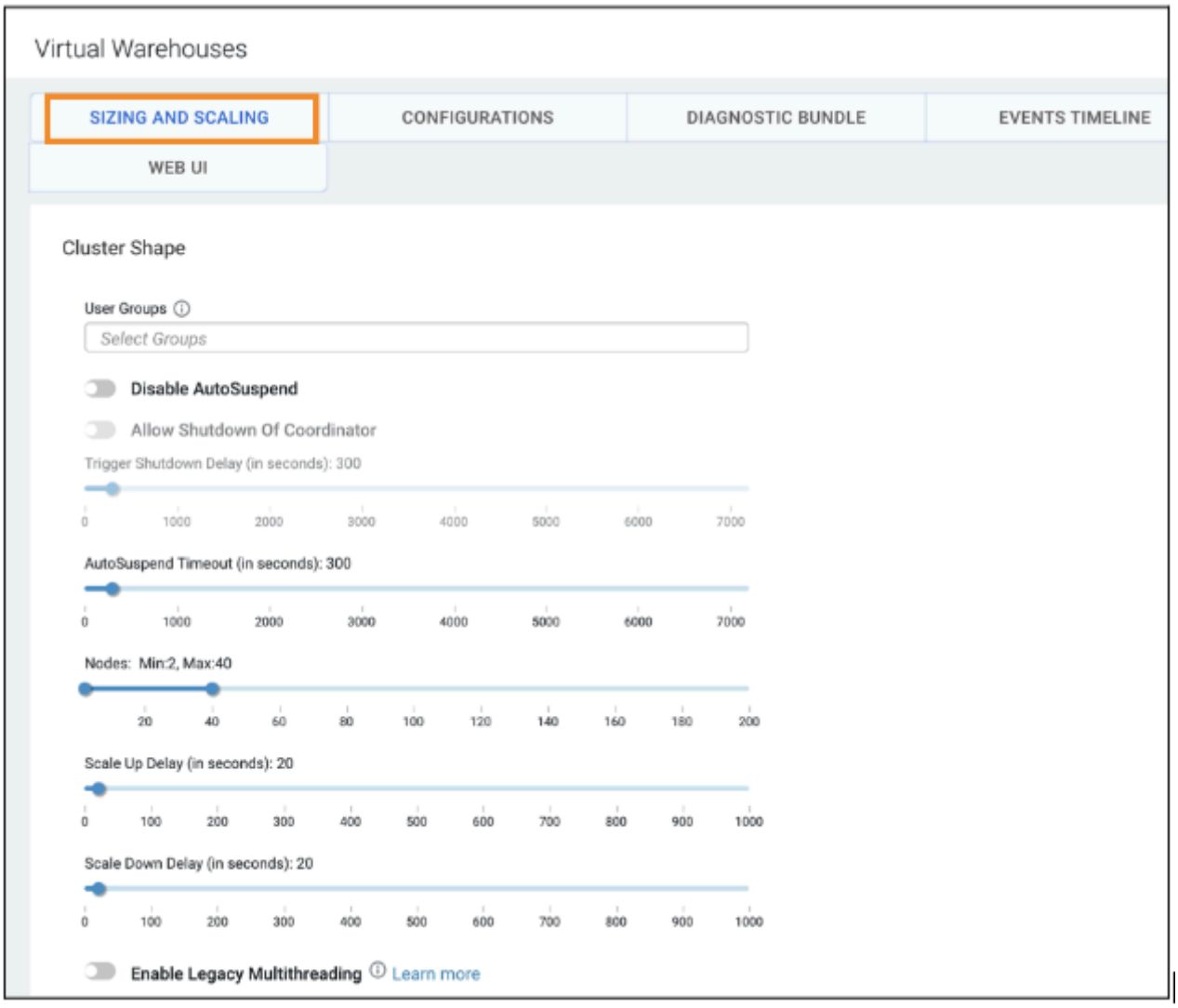
Queries executed using Impala are handled as follows:

1. User applications send SQL queries to Impala through ODBC or JDBC, which provide standardized querying interfaces. The user application may connect to any impalad in the cluster. This impalad becomes the coordinator for the query.
2. Impala parses the query and analyzes it to determine what tasks need to be performed by impalad instances across the cluster. Execution is planned for optimal efficiency.
3. Storage services are accessed by local impalad instances to provide data.
4. Each impalad returns data to the coordinating impalad, which sends these results to the client.

Components of Impala

The Impala service is a distributed, massively parallel processing (MPP) database engine. It consists of different daemon processes called as components. This topic describes the different roles these components play for a selected Virtual Warehouse.

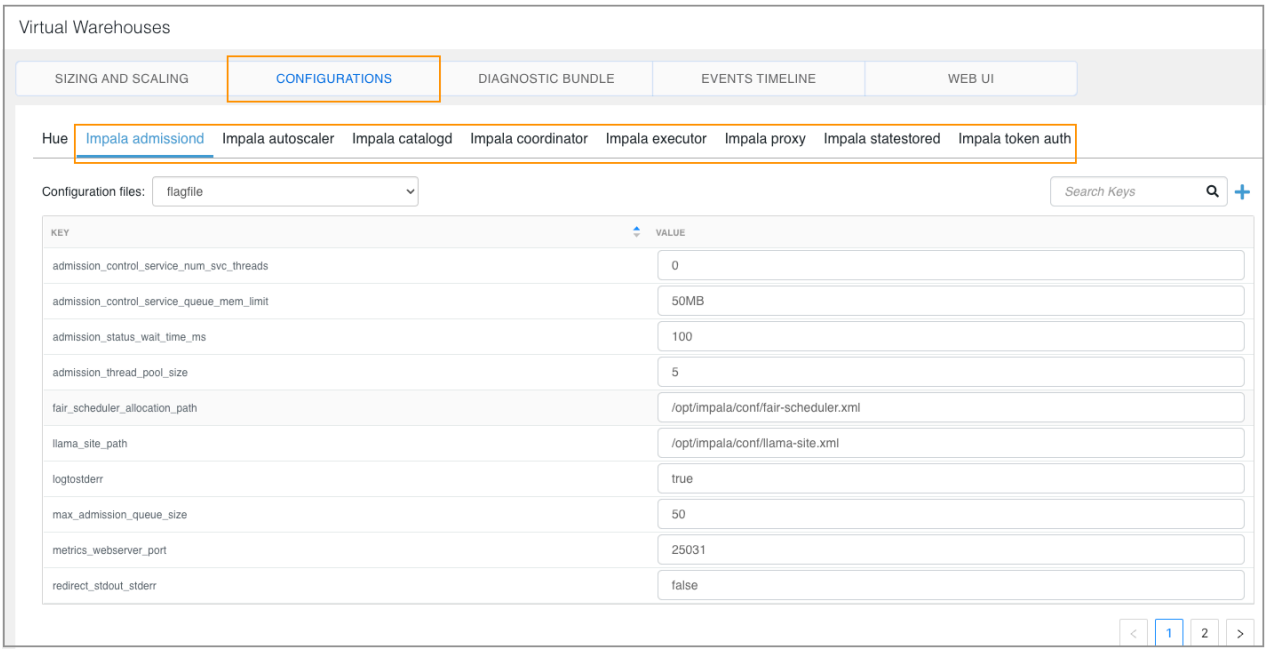
When you create an Impala Virtual Warehouse, it is automatically optimized for your workload by the Cloudera Data Warehouse service. Due to the containerized and compute-isolated architecture of Cloudera Data Warehouse, as well as intelligence to assign different default configurations, you do not have to customize your environment to optimize performance or to avoid resource usage spikes and out-of-memory conditions. However, if you must adjust some settings to suit your needs after creating an Impala Virtual Warehouse, you can add particular component configuration details in one of the free-form fields on the SIZING AND SCALING tab or under one of the components available under the CONFIGURATIONS tab for a selected Virtual Warehouse.



Before making any configuration changes, you can review the default configuration details in the components available under the CONFIGURATIONS tab in a Virtual Warehouse.

Impala service in Cloudera Data Warehouse consists of the following different daemon processes.

- Impala admissiond
- Impala catalogd
- Impala coordinator
- Impala executor
- Impala statestored
- Impala autoscaler
- Impala proxy



Components available for setting the configuration

This section provides reference information on supported configuration properties under the listed components. For the full list of configuration properties, see Impala Properties in Cloudera Runtime. Based on the information provided under the components, choose the appropriate component/role to tune a configuration if you must.

Note: Some of the configuration properties under these components can be tuned to your needs and some of the properties are not editable. In addition, the available properties may differ by Cloudera Data Warehouse Impala Runtime version.

Impala catalogd

The Catalog Server relays the metadata changes from Impala SQL statements to all the Impala daemons in a cluster. The catalog service avoids the need to issue REFRESH and INVALIDATE METADATA statements when the metadata changes are performed by statements issued through Impala.

Impala coordinator

A few of the key functions that an Impala coordinator performs are:

- Reads and writes to data files.
- Accepts queries transmitted from the impala-shell command, Hue, JDBC, or ODBC.
- Parallelizes the queries and distributes work across the cluster.

It is in constant communication with StateStore, to confirm which executors are healthy and can accept new work. Based on the health information it receives it assigns tasks to the executors. It also receives broadcast messages from the Catalog Server daemon whenever a cluster creates, alters, or drops any type of object, or when an INSERT or LOAD DATA statement is processed through Impala. It also communicates to Catalog Server daemon to load table metadata.

Impala executor

A few of the key functions that an Impala executor performs are:

- Executes the queries and transmits query results back to the central coordinator.
- Also transmits intermediate query results.

Depending on the size and the complexity of your queries you can select the number of executor nodes that are needed to run a typical query in your workloads. For more information on the

executor groups and recommendations on sizing your virtual warehouse to handle queries that must be run in your workloads concurrently, see Impala auto-scaling on public clouds.

Impala statestored

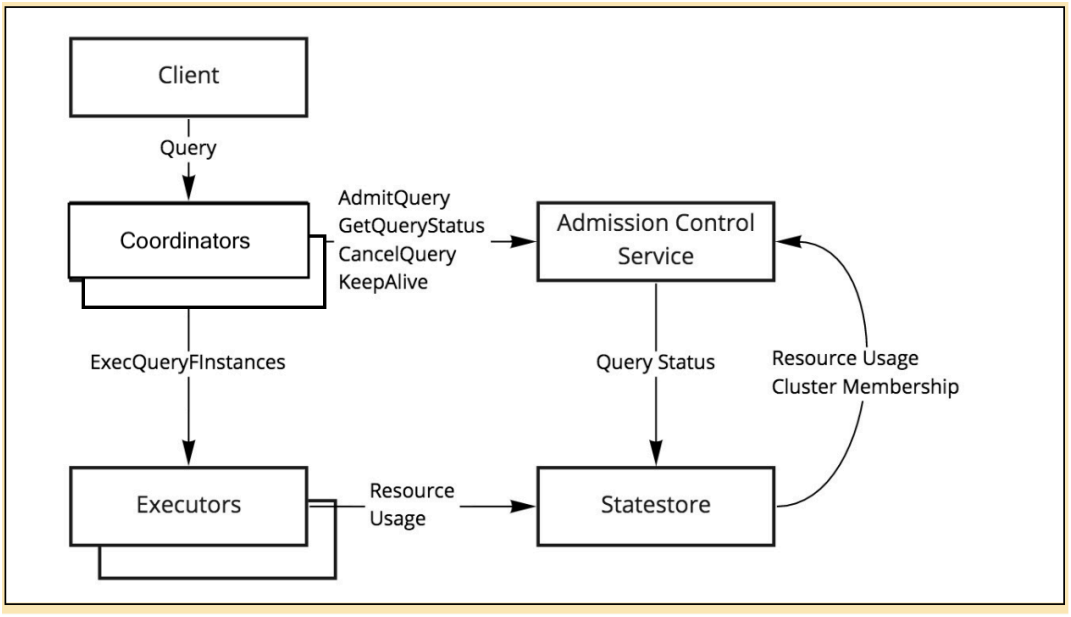
The Impala StateStore checks on the health of all Impala daemons in a cluster, and continuously relays its findings to each of those daemons. If an Impala daemon goes offline due to hardware failure, network error, software issue, or other reason, the StateStore informs all the other Impala daemons so that future queries can avoid making requests to the unreachable Impala daemon.

Impala admissiond

Before Cloudera Data Warehouse on premises 1.5.5 SP1 release, the admission controller is part of Impala coordinators. Each coordinator runs its local admission controller that uses eventually consistent information about the decisions made by other coordinators. In a multiple-coordinator setup, each coordinator with its local admission controller performs poorly because of the consistent nature of the admission decisions. Also, the traditional Impala multiple coordinator setups result in over admission of cluster resources.

To mitigate the above issue, Cloudera Data Warehouse on premises 1.5.5 SP1 a new service is added for admission control that runs in a separate process. This separation decouples its failure modes from coordinators and executors. However, the functionality is similar to the local admission controller implementation by scheduling queries for all executor groups and then attempting admission using round-robin algorithm. Separating the admission controller from the coordinator and running a single admission controller per cluster allows each coordinator to contact this service for each query to receive an admission decision.

Figure 1: Global admission controller



To impose limits on concurrent SQL queries to avoid resource usage spikes and out-of-memory conditions on busy Cloudera clusters the following configuration flags are set by default:

- The admission_control_service_queue_mem_limit flag specifies the limit on RPC payload consumption for the admission control service. The value is specified as number of bytes <int >[bB]?, megabytes <float>[mM], gigabytes <float>[gG], or percentage of the process memory limit <int>%. If no unit is specified, the default unit is bytes.
- The admission_control_service_num_svc_threads flag specifies the number of threads for processing the admission control service RPCs. If the default value, 0 is used, the value is set to the number of CPU cores.

- The admission_thread_pool_size flag specifies the size of the thread pool processing AdmitQuery requests.
- The max_admission_queue_size flag specifies the maximum size of the queue for the AdmitQuery thread pool.
- The admission_status_wait_time_ms flag specifies the time in milliseconds that the GetQueryStatus() RPC in the admission control service waits for the admission to complete before returning.

Adjusting the admission control configuration flags is possible.

Components available for diagnostic purpose only

The following components are read-only, and available only to view for diagnostic purposes. The properties listed under them are not tunable.

Impala autoscaler

One of the core Impala Virtual Warehouse components is Impala autoscaler. Impala autoscaler is in constant communication with coordinators and executors to determine when more or fewer compute resources are needed, given the incoming query load. When the autoscaler detects an imbalance in resources, it sends a request to the Kubernetes framework to increase or decrease the number of executor groups in the Virtual Warehouse, thereby right-sizing the amount of resources available for queries. This ensures that workload demand is met without wasting cloud resources.

Impala proxy

Impala Proxy is a small footprint reverse proxy that will forward every http client request to the Coordinator endpoint.

When you create an Impala warehouse with the 'allow coordinator shutdown' option a proxy 'impala proxy' is created to act as a connection endpoint to the impala clients. This option when enabled allows Impala coordinators to automatically shut down during idle periods. If the coordinator has shut down because of idle period, and if there is a request from a client when the coordinator is not running, impala proxy triggers the coordinator to start. So the proxy acts as a layer between the clients and the coordinator so that the clients connected to the VW do not time out when the coordinator starts up.