Cloudera Runtime 7.2.18

Apache Impala

Date published: 2020-11-30
Date modified: 2024-03-25

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 , 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 that run on specific hosts within your Hadoop cluster.

Impala service consists of the following categories of processes, referred as roles.

Impala Daemon

The core Impala component is the Impala daemon, physically represented by the impalad process. A few of the key functions that an Impala daemon performs are:

- Reads and writes to data files.
- Accepts queries transmitted from the impala-shell command, , JDBC, or ODBC.
- Parallelizes the queries and distributes work across the cluster.
- Transmits intermediate query results back to the central coordinator.

Impala daemons can be deployed in one of the following ways:

- HDFS and Impala are co-located, and each Impala daemon runs on the same host as a DataNode.
- Impala is deployed separately in a compute cluster and reads remotely from HDFS, S3, ADLS, etc.

The Impala daemons are in constant communication with StateStore, to confirm which daemons are healthy and can accept new work.

They also receive broadcast messages from the Catalog Server daemon whenever an Impala daemon in the cluster creates, alters, or drops any type of object, or when an INSERT or LOAD DATA statement is processed through Impala. This background communication minimizes the need for REFRESH or INVALIDATE METADATA statements that were needed to coordinate metadata across Impala daemons.

You can control which hosts act as query coordinators and which act as query executors, to improve scalability for highly concurrent workloads on large clusters.

Note: Impala Daemons should be deployed on nodes using the same Glibc version since different Glibc version supports different Unicode standard version and also ensure that the en_US.UTF-8 locale is installed in the nodes. Not using the same Glibc version might result in inconsistent UTF-8 behavior when UTF8_MODE is set to true.

Impala StateStore

The Impala StateStore checks on the health of all Impala daemons in a cluster, and continuously relays its findings to each of those daemons. It is physically represented by a daemon process named statestored. You only need such a process on one host in a cluster. 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.

Because the StateStore's purpose is to help when things go wrong and to broadcast metadata to coordinators, it is not always critical to the normal operation of an Impala cluster. If the StateStore is not running or becomes unreachable, the Impala daemons continue running and distributing work among themselves as usual when working with the data known to Impala. The cluster just becomes less robust if other Impala daemons fail, and metadata becomes less consistent as it changes while the StateStore is offline. When the StateStore comes back online, it re-establishes communication with the Impala daemons and resumes its monitoring and broadcasting functions.

If you issue a DDL statement while the StateStore is down, the queries that access the new object the DDL created will fail.

Impala Catalog Server

The Catalog Server relays the metadata changes from Impala SQL statements to all the Impala daemons in a cluster. It is physically represented by a daemon process named catalogd. You only need such a process on one host in a cluster. Because the requests are passed through the StateStore daemon, it makes sense to run the `statestored` and `catalogd` services on the same host.

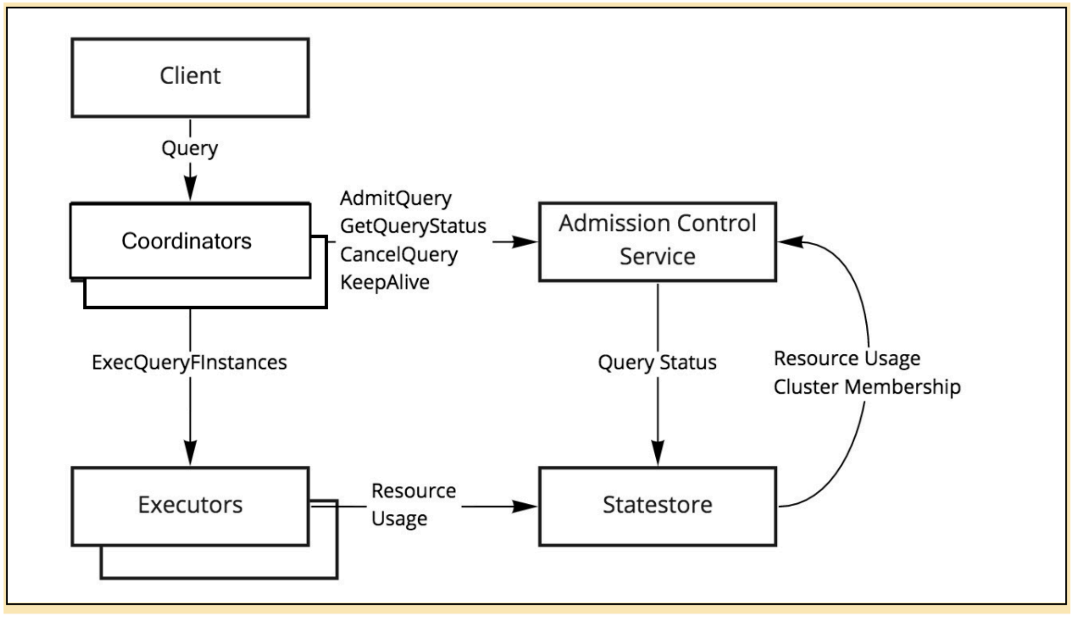
The catalog service avoids the need to issue REFRESH and INVALIDATE METADATA statements when the metadata changes are performed by statements issued through Impala. When you create a table, load data, and so on through Hive, you do need to issue REFRESH or INVA LIDATE METADATA on an Impala node before executing a query there.

Impala admissiond

Before Cloudera Runtime 7.3.2 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, 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.

If a single admissiond service runs for the entire cluster, what happens if it goes down? When an admissiond service becomes unavailable, the coordinator starts a retry loop to reconnect and submit the query. This retrying behavior continues until the configured limit, admission_max_retry_time_s, is reached. This process attempts to cover short, transient network outages. If the coordinator cannot establish a successful connection before the admission_max_retry_time_s is reached, the query fails with a network-related error, such as:

```
ERROR: Query <query_ID> failed: Could not find IPv4 address for:
 admissiond
```

When the admissiond service recovers, there is a short period where the DNS records are updated. During this time, new queries may still fail as coordinators are temporarily unable to resolve the service's IP address. This issue is self-resolving, and queries will succeed once the DNS changes propagate and coordinators can connect to the recovered admissiond service