Cloudera Runtime 7.2.7

Apache Hive Overview

Date published: 2019-08-21
Date modified: 2021-02-04

CLOUDƎRA

Legal Notice

Contents

Apache Hive features

Major changes to Apache Hive 2.x improve Apache Hive 3.x transactions and security. Knowing the major differences between these versions is critical for SQL users, including those who use Apache Spark and Apache Impala.

Hive is a data warehouse system for summarizing, querying, and analyzing huge, disparate data sets.

ACID transaction processing

Hive 3 tables are ACID (Atomicity, Consistency, Isolation, and Durability)-compliant. Hive 3 write and read operations improve the performance of transactional tables. Atomic operations include simple writes and inserts, writes to multiple partitions, and multiple inserts in a single SELECT statement. A read operation is not affected by changes that occur during the operation. You can insert or delete data, and it remains consistent throughout software and hardware crashes. Creation and maintenance of Hive tables is simplified because there is no longer any need to bucket tables.

Materialized views

Because multiple queries frequently need the same intermediate roll up or joined table, you can avoid costly, repetitious query portion sharing, by precomputing and caching intermediate tables into views.

Query results cache

Hive filters and caches similar or identical queries. Hive does not recompute the data that has not changed. Caching repetitive queries can reduce the load substantially when hundreds or thousands of users of BI tools and web services query Hive.

Scheduled Queries

Using SQL statements, you can schedule Hive queries to run on a recurring basis, monitor query progress, temporarily ignore a query schedule, and limit the number running in parallel. You can use scheduled queries to start compaction and periodically rebuild materialized views, for example.

Spark integration with Hive in Data Hub

Spark and Hive tables interoperate using the Hive Warehouse Connector and Spark Direct Reader to access ACID managed tables. You can access external tables from Spark directly using SparkSQL.

You do not need HWC to read or write Hive external tables. Spark users just read from or write to Hive directly. You can read Hive external tables in ORC or Parquet formats. You can write Hive external tables in ORC format only.

Security improvements

Apache Ranger secures Hive data by default. To meet demands for concurrency improvements, ACID support, render security, and other features, Hive tightly controls the location of the warehouse on a file system, or object store, and memory resources.

With Apache Ranger and Apache Hive ACID support, your organization will be ready to support and implement GDPR (General Data Protection Regulation).

Connection Pooling

Hive supports HakariCP JDBC connection pooling.

Unsupported features

CDP does not support the following features that were available in HDP and CDH platforms:

- CREATE TABLE that specifies a managed table location

  Do not use the LOCATION clause to create a managed table. Hive assigns a default location in the warehouse to managed tables.
- CREATE INDEX

  Hive builds and stores indexes in ORC or Parquet within the main table, instead of a different table, automatically. Set hive.optimize.index.filter to enable use (not recommended--use materialized views instead). Existing indexes are preserved and migrated in Parquet or ORC to CDP during upgrade.

Hive on Tez introduction

The Cloudera Data Platform (CDP) service that provides an Apache Hive SQL database that Apache Tez executes.

The Hive on Tez service provides a SQL-based data warehouse system based on Apache Hive 3.x. The enhancements in Hive 3.x over previous versions can improve SQL query performance, security, and auditing capabilities. The Hive metastore (HMS) is a separate service, not part of Hive, not even necessarily on the same cluster. HMS stores the metadata on the backend for Hive, Impala, Spark, and other components.

Apache Tez is the Hive execution engine for the Hive on Tez service, which includes HiveServer (HS2) in Cloudera Manager. MapReduce is not supported. In a Cloudera cluster, if a legacy script or application specifies MapReduce for execution, an exception occurs. Most user-defined functions (UDFs) require no change to execute on Tez instead of MapReduce.

With expressions of directed acyclic graphs (DAGs) and data transfer primitives, execution of Hive queries on Tez instead of MapReduce improves query performance. In Cloudera Data Platform (CDP), Tez is usually used only by Hive, and launches and manages Tez AM automatically when Hive on Tez starts. SQL queries you submit to Hive are executed as follows:

- Hive compiles the query.
- Tez executes the query.
- Resources are allocated for applications across the cluster.
- Hive updates the data in the data source and returns query results.

Hive on Tez runs tasks on ephemeral containers and uses the standard YARN shuffle service. By default, Hive data is stored on HDFS. If you do not enable the Ranger security service, or other security, by default Hive uses storage-based authorization (SBA) based on user impersonation.

Hive low-latency analytical processing

CDP Public Cloud supports low-latency analytical processing (LLAP) of Hive queries. Using LLAP in the CDP Data Warehouse service, you can tune your data warehouse infrastructure, components, and client connection parameters to improve the performance and relevance of business intelligence and other applications.

Increasingly, enterprises want to run SQL workloads that return faster results than batch processing can provide. These enterprises often want data analytics applications to support interactive queries. Low-latency analytical processing (LLAP) can improve the performance of interactive queries. A Hive interactive query that runs on the CDP Public Cloud meets low-latency, variably guaged benchmarks to which Hive LLAP responds in 15 seconds or less. LLAP enables application development and IT infrastructure to run queries that return real-time or near-real-time results.

LLAP caches data for multiple queries and this capability does not support user impersonation.

Hive unsupported interfaces and features in public clouds

You need to know the interfaces available in HDP or CDH platforms that are not supported in CDP.

The following interfaces are not supported:

- Hcat CLI
- Hive CLI (replaced by Beeline)
- Hive View UI feature in Ambari
- MapReduce execution engine (replaced by LLAP)
- Pig
- Spark execution engine
- Spark thrift server
- SQL Standard Authorization
- Tez View UI feature in Ambari
- WebHCat

You can use Hue in lieu of Hive View.

Unsupported Features

CDP does not support the following features that were available in HDP and CDH platforms:

- CREATE TABLE that specifies a managed table location

  Do not use the LOCATION clause to create a managed table. Hive assigns a default location in the warehouse to managed tables. That default location is configured in Hive using the hive.metastore.warehouse.dir configuration property, but can be overridden for the database by setting the CREATE DATABASE MANAGEDLOCATION parameter.

- CREATE INDEX and related index commands were removed in Hive 3, and consequently are not supported in CDP.

  In CDP, you use the Hive 3 default ORC columnar file formats to achieve the performance benefits of indexing. Materialized Views with automatic query rewriting also improves performance. Indexes migrated to CDP are preserved but render any Hive tables with an undroppable index. To drop the index, google the Known Issue for CDPD-23041.

- Hive metastore (HMS) high availablility (HA) load balancing in CDH

  You need to set up HMS HA as described in the documentation.

- Local or Embedded Hive metastore server

  CDP does not support the use of a local or embedded Hive metastore setup.

Unsupported Connector Use

CDP does not support the Sqoop exports using the Hadoop jar command (the Java API) that Teradata documents. For more information, see Migrating data using Sqoop.

Apache Hive 3 architectural overview

Understanding Apache Hive 3 major design features, such as default ACID transaction processing, can help you use Hive to address the growing needs of enterprise data warehouse systems.

Data storage and access control

One of the major architectural changes to support Hive 3 design gives Hive much more control over metadata memory resources and the file system, or object store. The following architectural changes from Hive 2 to Hive 3 provide improved security:

- Tightly controlled file system and computer memory resources, replacing flexible boundaries: Definitive boundaries increase predictability. Greater file system control improves security.
- Optimized workloads in shared files and YARN containers

Hive 3 is optimized for object stores in the following ways:

- Hive uses ACID to determine which files to read rather than relying on the storage system.
- In Hive 3, file movement is reduced from that in Hive 2.
- Hive caches metadata and data agressively to reduce file system operations

The major authorization model for Hive is Ranger. Hive enforces access controls specified in Ranger. This model offers stronger security than other security schemes and more flexibility in managing policies.

This model permits only Hive to access the Hive warehouse.

Transaction processing

You can deploy new Hive application types by taking advantage of the following transaction processing characteristics:

- Mature versions of ACID transaction processing:

  ACID tables are the default table type.

  ACID enabled by default causes no performance or operational overload.
- Simplified application development, operations with strong transactional guarantees, and simple semantics for SQL commands

  You do not need to bucket ACID tables.
- Materialized view rewrites
- Automatic query cache
- Advanced optimizations

Hive client changes

You can use the thin client Beeline for querying Hive from the command line. You can run Hive administrative commands from the command line. Beeline uses a JDBC connection to Hive to execute commands. Hive parses, compiles, and execute operations. Beeline supports many of the command-line options that Hive CLI supported. Beeline does not support hive -e set   key=value to configure the Hive Metastore.

You enter supported Hive CLI commands by invoking Beeline using the hive keyword, command option, and command. For example, hive -e set. Using Beeline instead of the thick client Hive CLI, which is no longer supported, has several advantages, including low overhead. Beeline does not use the entire Hive code base. A small number of daemons required to execute queries simplifies monitoring and debugging.

Hive enforces whitelist and blacklist settings that you can change using SET commands. Using the blacklist, you can restrict memory configuration changes to prevent instability. Different Hive instances with different whitelists and blacklists to establish different levels of stability.

Apache Hive Metastore sharing

Hive, Impala, and other components can share a remote Hive metastore.

Spark integration

Spark and Hive tables interoperate using the Hive Warehouse Connector.

You can use the Hive Warehouse Connector (HWC) to access Hive managed tables from Spark. HWC is specifically designed to access managed Hive tables, and supports writing to tables in ORC format only.

You do not need HWC to read from or write to Hive external tables. Spark uses native Spark to read external tables.

Query execution of batch and interactive workloads

You can connect to Hive using a JDBC command-line tool, such as Beeline, or using an JDBC/ODBC driver with a BI tool, such as Tableau. You configure the settings file for each instance to perform either batch or interactive processing.

Installing Hive on Tez and adding a HiveServer role

Cloudera Runtime (CR) services include Hive on Tez and Hive Metastore (HMS). Hive on Tez is a SQL query engine using Apache Tez that performs the HiveServer (HS2) role in a Cloudera cluster. You need to install Hive on Tez and HMS in the correct order; otherwise, HiveServer fails. You need to install additional HiveServer roles to Hive on Tez, not the Hive service; otherwise, HiveServer fails.

Procedure

1. Add the Hive service to a cluster.

> ⚠️ **Warning:** Do not add the HiveServer2 role to the Hive service. Only the Hive on Tez service supports this role.

2. Add the Hive on Tez service to the same cluster.

The Hive on Tez service includes the HiveServer2 role.

3. Accept the default, or change the Hive warehouse location for managed and external tables as described below.

Adding a HiveServer role

Procedure

1. In Cloudera Manager, click Clusters Hive on Tez .

Do not click Clusters Hive by mistake. Only the Hive on Tez service supports the HiveServer2 role.

2. Click Actions Add Role Instances .

3. Click in the HiveServer2 box to select hosts.

4. In the Host name column, select a host for the HiveServer2 role, and click OK.

The selected host name you assigned the HiveServer2 role appears under HiveServer2.

5. Click Continue.

The new HiveServer2 role state is stopped.

6. Select the new HiveServer2 role.

7. In Actions for Selected, select Start, and then click Start to confirm.

You see that the service successfully started.



Changing the Hive warehouse location

About this task

You use the Hive Metastore Action menu in Cloudera Manager, and navigate to one of the following menu items in the first step below.

- Hive Action Menu Create Hive Warehouse Directory
- Hive Action Menu Create Hive Warehouse External Directory

Procedure

1. Set up directories for the Hive warehouse directory and Hive warehouse external directory from Cloudera Manager Actions.
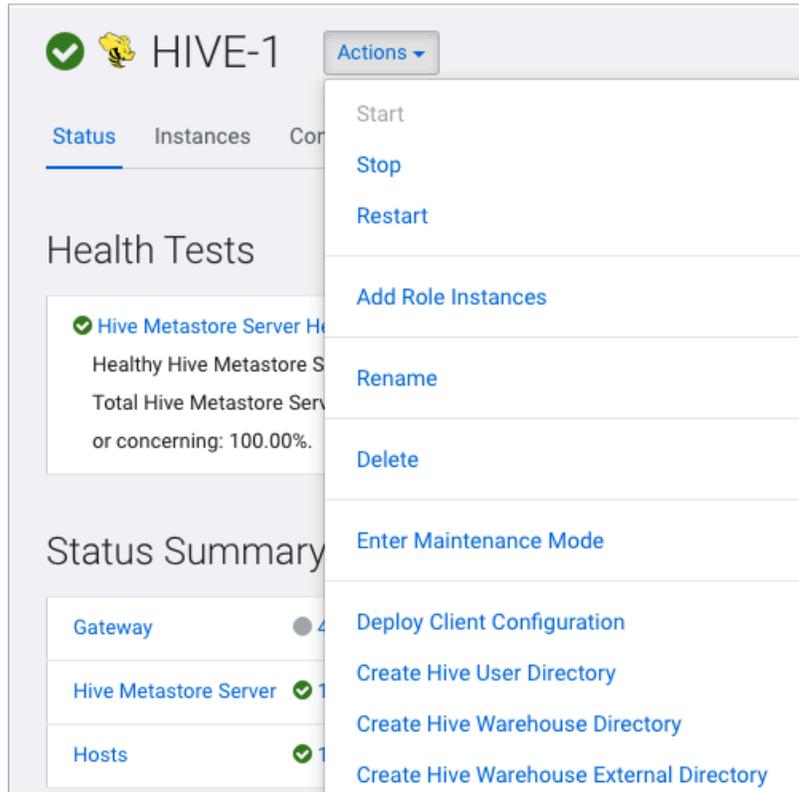
2. In Cloudera Manager, click Clusters Hive (the Hive Metastore service) Configuration , and change the hive.metastore.warehouse.dir property value to the path you specified for the new Hive warehouse directory.

3. Change the hive.metastore.warehouse.external.dir property value to the path you specified for the Hive warehouse external directory.

4. Configure Ranger policies or set up ACL permissions to access the directories.

Apache Hive content roadmap

The content roadmap provides links to the available content resources for Apache Hive.

Table 1: Apache Hive Content roadmap

Task	Resources	Source	Description
Understanding	Presentations and Papers about Hive	Apache wiki	Contains meeting notes, presentations, and whitepapers from the Apache community.
Getting Started	Hive Tutorial	Apache wiki	Provides a basic overview of Apache Hive and contains some examples on working with tables, loading data, and querying and inserting data.
Developing	Materialized Views	Apache wiki	Covers accelerating query processing in data warehouses by pre-computing summaries using materialized views.
	JdbcStorageHandler	Apache wiki	Describes how to read from a JDBC data source in Hive.
	Hive transactions	Apache wiki	Describes ACID operations in Hive.

Task	Resources	Source	Description
	Hive Streaming API	Apache wiki	Explains how to use an API for pumping data continuously into Hive using clients such as NiFi and Flume.
	Hive Operators and Functions	Apache wiki	Describes the Language Manual UDF.
	Beeline: HiveServer2 Client	Apache wiki	Describes how to use the Beeline client.
Reference	SQL Language Manual	Apache wiki	Language reference documentation available in the Apache wiki.
Contributing	Hive Developer FAQ	Apache wiki	Resources available if you want to contribute to the Apache community.
	How to Contribute	Apache wiki	
	Hive Developer Guide	Apache wiki	
	Plug-in Developer Kit	Apache wiki	
	Unit Test Parallel Execution	Apache wiki	
	Hive Architecture Overview	Apache wiki	
	Hive Design Docs	Apache wiki	
	Project Bylaws	Apache wiki	