Data Access 3

Using Druid and Apache Hive

Date of Publish: 2018-07-12

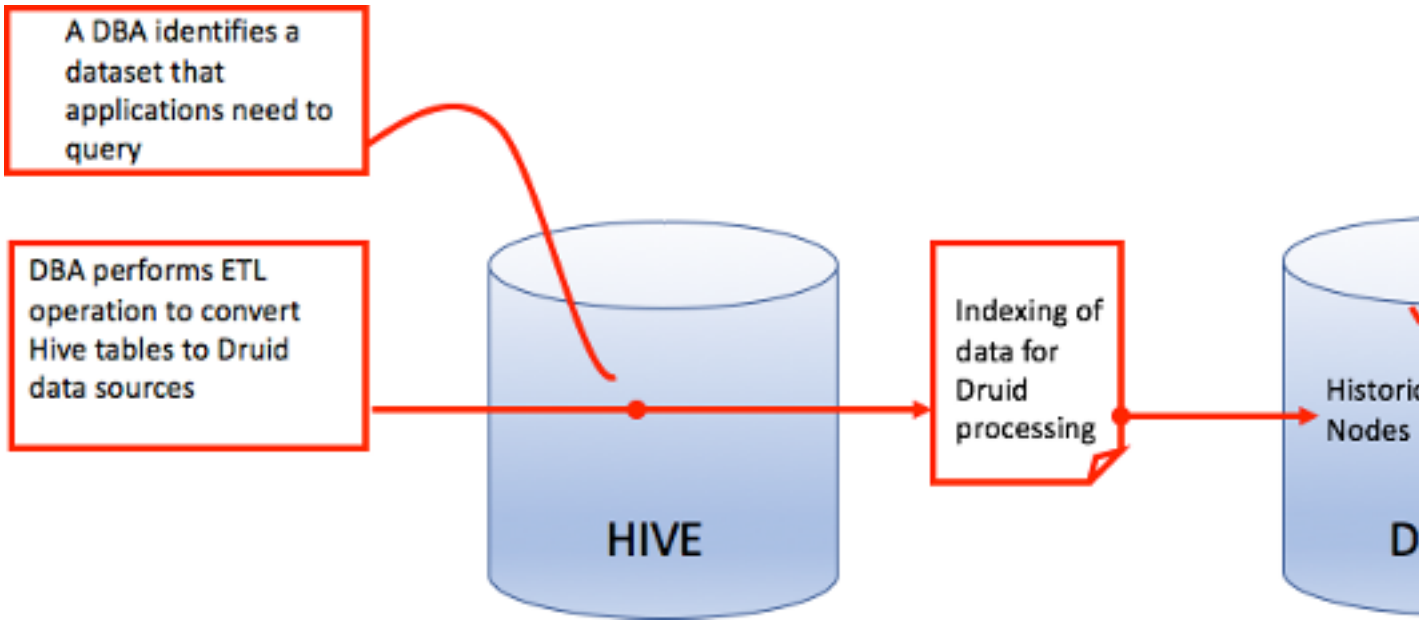
http://docs.hortonworks.com

Contents

Accelerating Hive queries using Druid

You can perform interactive analytic queries on real-time and historical data using the HDP integration of Hive and Apache Druid (incubating). You can discover existing Druid data sources as external tables, create or ingest batch data into Druid, set up Druid-Kafka streaming ingestion using Hive, and query Druid data sources from Hive.

The integration of Hive with Druid places a SQL layer on Druid. After Druid ingests data from a Hive enterprise data warehouse (EDW), the interactive and sub-second query capabilities of Druid can be used to accelerate queries on historical data from the EDW. Hive integration with Druid enables applications such as Tableau to scale while queries run concurrently on both real-time and historical data. The following figure is an overview of how Hive historical data can be brought into a Druid environment. Queries analyzing Hive-sourced data are run directly on the historical nodes of Druid after indexing between the two databases completes.



Related Information

Druid Integration

How Druid indexes Hive data

Before you can create a Druid data source based on Hive data, you must understand how Hive external table data maps to the column orientation and segment files of Druid.

Mapping of a Hive external table to a Druid file

Each Druid segment consists of the following objects to facilitate fast lookup and aggregation:

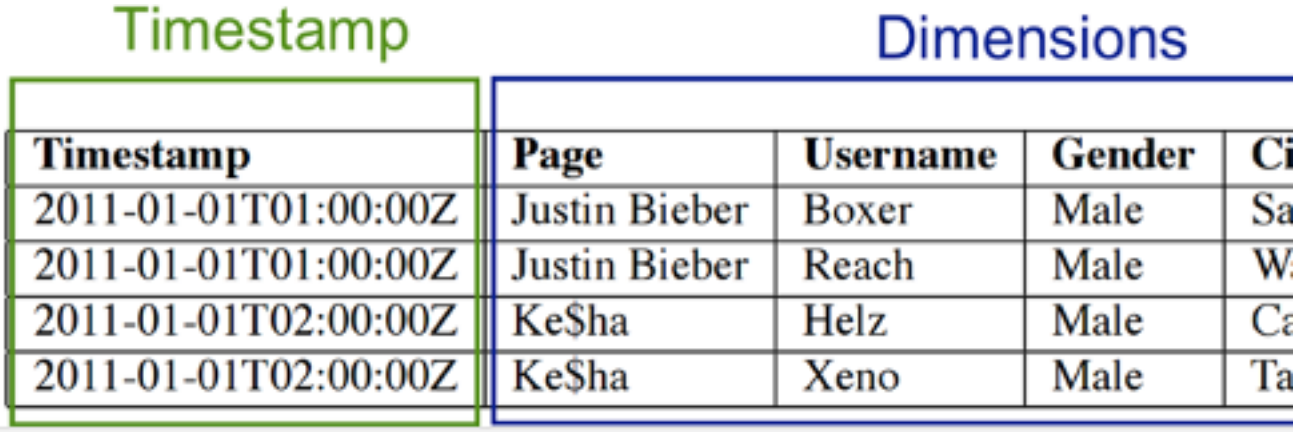
Timestamp column	The SQL-based timestamp column is filled in based on how you set the time granularity of imported Hive data and what time range of data is selected in the Hive external table. This column is essential for indexing the data in Druid because Druid itself is a time-series database. The timestamp column must be named __time.
Dimension columns	The dimension columns are used to set string attributes for search and filter operations. To index a Hive-sourced

column as a Druid dimension column, you must cast the column as a string type.

Metric columns

Metric columns are used to index metrics for use as aggregates or measures. To index a Hive-sourced column as a Druid metric column, you must cast the column as a Hive numeric data type.

The following figure shows how you can categorize Druid data into three types of columns.



Related Information

Druid Integration

Transform Apache Hive Data to Druid

You can execute a Hive query to transform data in Hive to a data source in Druid.

Before you begin

- If you use Kerberos, configure and run Hive low-latency analytical processing (LLAP).
- You set up a table, ssb_10_flat_orc as defined in the Star Schema Benchmark example.

About this task

A Hive SQL command, an excerpt from the Star Schema Benchmark using the Hive/Druid Integration, invokes the Druid storage handler, specifies Druid segment granularity, and maps selected Hive columns to Druid column types.

Procedure

1. Put all the Hive data to undergo ETL in a Hive table.
2. Execute a Hive query to set the location of the Druid broker using a DNS name or IP address and port 8082, the default broker text listening port.

   ```
   SET hive.druid.broker.address.default=10.10.20.30:8082;
   ```

   If you installed the Hive and Druid services using Ambari, you can skip this step.
3. Run a CREATE TABLE AS SELECT statement to create a new Druid datasource from the Hive table you selected in step 1.
   The following example pushes Hive data to Druid.

   ```
   CREATE TABLE ssb_druid_hive
   ```

```
STORED BY 'org.apache.hadoop.hive.
druid.DruidStorageHandler'
TBLPROPERTIES (
"druid.segment.granularity" = "MONTH",
"druid.query.granularity" = "DAY")
AS
SELECT
cast(d_year || '-' || d_monthnuminyear || '-' || d_daynuminmonth as
 timestamp) as `__time`,
cast(c_city as string) c_city,
cast(c_nation as string) c_nation,
cast(c_region as string) c_region,
cast(d_weeknuminyear as string) d_weeknuminyear,
cast(d_year as string) d_year,
cast(d_yearmonth as string) d_yearmonth,
cast(d_yearmonthnum as string) d_yearmonthnum,
cast(lo_discount as string) lo_discount,
cast(lo_quantity as string) lo_quantity,
cast(p_brand1 as string) p_brand1,
cast(p_category as string) p_category,
cast(p_mfgr as string) p_mfgr,
cast(s_city as string) s_city,
cast(s_nation as string) s_nation,
cast(s_region as string) s_region,
lo_revenue,
lo_extendedprice * lo_discount discounted_price,
lo_revenue - lo_supplycost net_revenue
FROM
ssb_10_flat_orc.customer, ssb_10_flat_orc.dates,
 ssb_10_flat_orc.lineorder,
ssb_10_flat_orc.part, ssb_10_flat_orc.supplier
where
lo_orderdate = d_datekey and lo_partkey = p_partkey
and lo_suppkey = s_suppkey and lo_custkey = c_custkey;
```

Related Information

Druid Integration

Anatomy of a Hive-to-Druid data transformation

A breakdown of the example SQL code that transforms Hive data into a Druid datasource helps you understand how to transform your own data.

SQL example code

The following SQL statement contains the main elements of a statement that can transform Hive data into a time series-based Druid datasource. You need to replace the values in the statement to match your data warehouse environment and analytics parameters.

```
CREATE TABLE ssb_druid_hive
STORED BY 'org.apache.hadoop.hive.
druid.DruidStorageHandler'
TBLPROPERTIES (
"druid.segment.granularity" = "MONTH",
"druid.query.granularity" = "DAY")
AS
SELECT
cast(d_year || '-' || d_monthnuminyear || '-' || d_daynuminmonth as
 timestamp) as `__time`,
cast(c_city as string) c_city,
cast(c_nation as string) c_nation,
```

```
cast(c_region as string) c_region,
cast(d_weeknuminyear as string) d_weeknuminyear,
cast(d_year as string) d_year,
cast(d_yearmonth as string) d_yearmonth,
cast(d_yearmonthnum as string) d_yearmonthnum,
cast(lo_discount as string) lo_discount,
cast(lo_quantity as string) lo_quantity,
cast(p_brand1 as string) p_brand1,
cast(p_category as string) p_category,
cast(p_mfgr as string) p_mfgr,
cast(s_city as string) s_city,
cast(s_nation as string) s_nation,
cast(s_region as string) s_region,
lo_revenue,
lo_extendedprice * lo_discount discounted_price,
lo_revenue - lo_supplycost net_revenue
FROM
ssb_10_flat_orc.customer, ssb_10_flat_orc.dates, ssb_10_flat_orc.lineorder,
ssb_10_flat_orc.part, ssb_10_flat_orc.supplier
where
lo_orderdate = d_datekey and lo_partkey = p_partkey
and lo_suppkey = s_suppkey and lo_custkey = c_custkey;
```

Explanation of SQL example

The following breakdown of the preceding SQL statement explains the main elements of a statement that can transform Hive data into a time series-based Druid datasource. You need to replace the values in the statement to match your data warehouse environment and analytics parameters.

CREATE TABLE ssb_druid_hive	Creates the Hive table and assigns a name to it. You must use a table name that is not already used by another Druid datasource.
STORED BY, 'org.apache.hadoop.hive., druid.DruidStorageHandler'	This calls the Druid storage handler so that the Hive data can be transformed to a Druid datasource.
TBLPROPERTIES ("druid.segment.granularity" = "MONTH", "druid.query.granularity" = "DAY") AS SELECT cast(d_year || '-' || d_monthnuminyear || '-' || d_daynuminmonth as timestamp) as `__time`,	Creates the __time column, which is a required SQL timestamp column for the Druid datasource.
cast(c_city as string) c_city, cast(c_nation as string) c_nation, cast(c_region as string) c_region, cast(d_weeknuminyear as string) d_weeknuminyear, ...	cast (… as string) statements index columns as dimensions. Dimensions in the Druid datasource are used to search and filter.
lo_extendedprice * lo_discount discounted_price, lo_revenue - lo_supplycost net_revenue	These lines preaggregate metrics columns. To index a column as metrics, you need to cast the column to a Hive numeric data type.
FROM ssb_10_flat_orc.customer, ssb_10_flat_orc.dates, ...	The numeric value indicates the data scale. The other information corresponds with the name and other components of the source Hive tables.

Table Property	Required	Description	Valid Values
druid.segment.popularity	No	Defines how the data is physically partitioned. The values that are permissible here correspond with Druid segment granularity.	"YEAR", "MONTH", "WEEK", "DAY", "HOUR", "MINUTE", "SECOND"

Table Property	Required	Description	Valid Values
druid.query.granularity	No	Defines how much granularity to store in a segment. The values that are permissible here correspond with Druid query granularity.	"YEAR", "MONTH", "WEEK", "DAY", "HOUR", "MINUTE", "SECOND"

If you need Druid to ingest Hive data that follows the same schema as the first data set that you transformed, you can do so with the INSERT INTO statement.

Create a Hive materialized view, store it in Druid

You can create a materialized view and store it in Druid using the Druid storage handler.

Before you begin

- Hive is running as a service in the cluster.
- Druid is running as a service in the cluster.
- You created a transactional table named src that has timestamp, dimension, and metric columns: __time TIMESTAMP, page STRING, user STRING, c_added INT, and c_removed INT columns.

About this task

In this task, you include the STORED BY clause followed by the Druid storage handler. The storage handler integrates Hive and Druid for saving the materialized view in Druid.

Procedure

1. Execute a Hive query to set the location of the Druid broker using a DNS name or IP address and port 8082, the default broker text listening port.

   ```
   SET hive.druid.broker.address.default=10.10.20.30:8082;
   ```

   If you installed the Hive and Druid services using Ambari, you can skip this step.
2. Create a materialized view store the view in Druid.

   ```
   CREATE MATERIALIZED VIEW druid_mv
     STORED BY 'org.apache.hadoop.hive.druid.DruidStorageHandler'
     AS SELECT __time, page, user, c_added, c_removed
     FROM src;
   ```

Related Information

Druid Integration

Druid and Hive tuning

As administrator, you can set druid.hive properties to improve Druid-Hive performance.

Performance related druid.hive properties

If Hive and Druid are installed with Ambari, the properties are set and tuned for your cluster automatically. However, you can fine-tune some properties if you detect performance problems with applications that are running the queries. The following list includes some of the Druid properties that can be used by Hive. As an HDP administrator, you can troubleshoot and customize a Hive-Druid integration using these properties.

Property	Description
hive.druid.indexer.segments.granularity	Granularity of the segments created by the Druid storage handler.
hive.druid.indexer.partition.size.max	Maximum number of records per segment partition.
hive.druid.indexer.memory.rownum.max	Maximum number of records in memory while storing data in Druid.
hive.druid.broker.address.default	Address of the Druid broker node. When Hive queries Druid, this address must be declared.
hive.druid.coordinator.address.default	Address of the Druid coordinator node. It is used to check the load status of newly created segments.
hive.druid.select.threshold	When a SELECT query is split, this is the maximum number of rows that Druid attempts to retrieve.
hive.druid.http.numConnection	Number of connections used by the HTTP client.
hive.druid.http.read.timeout	Read timeout period for the HTTP client in ISO8601 format. For example, P2W, P3M, PT1H30M, PT0.750S are possible values.
hive.druid.sleep.time	Sleep time between retries in ISO8601 format.
hive.druid.basePersistDirectory	Local temporary directory used to persist intermediate indexing state.
hive.druid.storage.storageDirectory	Deep storage location of Druid.
hive.druid.metadata.base	Default prefix for metadata table names.
hive.druid.metadata.db.type	Metadata database type. The only valid values are "mysql" and "postgresql"
hive.druid.metadata.uri	URI to connect to the database.
hive.druid.working.directory	Default HDFS working directory used to store some intermediate metadata.
hive.druid.maxTries	Maximum number of retries to connect to Druid before throwing an exception.
hive.druid.bitmap.type	Encoding algorithm use to encode the bitmaps.

If you installed both Hive and Druid with Ambari, then do not change any of the hive.druid.* properties other than those above when there are performance issues.

Related Information

Druid Integration