Kudu Integration

Hive Kudu Integration

Overview

Apache Kudu is a an Open Source data storage engine that makes fast analytics on fast and changing data easy.

Implementation

The initial implementation was added to Hive 4.0 in HIVE-12971 and is designed to work with Kudu 1.2+.

There are two main components which make up the implementation: the `KuduStorageHandler` and the `KuduPredicateHandler`. The `KuduStorageHandler` is a Hive StorageHandler implementation. The primary roles of this class are to manage the mapping of a Hive table to a Kudu table and configures Hive queries. The KuduPredicateHandler is used push down filter operations to Kudu for more efficient IO.

NOTE: The initial implementation is considered ***`experimental`*** as there are remaining sub-jiras open to make the implementation more configurable and performant. Currently only external tables pointing at existing Kudu tables are supported. Support for creating and altering underlying Kudu tables in tracked via HIVE-22021. Additionally full support for UPDATE, UPSERT, and DELETE statement support is tracked by HIVE-22027.

Hive Configuration

To issue queries against Kudu using Hive, one optional parameter can be provided by the Hive configuration:

Hive Configuration	
`hive.kudu.master.addresses.default`	Comma-separated list of all of the Kudu master addresses. This value is only used for a given table if the *kudu.master_addresses* table property is not set.

For those familiar with Kudu, the master addresses configuration is the normal configuration value necessary to connect to Kudu. The easiest way to provide this value is by using the `-hiveconf` option to the `hive` command.

```
hive -hiveconf hive.kudu.master.addresses.default=localhost:7051
```

Table Creation

To access Kudu tables, a Hive table must be created using the `CREATE` command with the `STORED BY` clause. Until HIVE-22021 is completed, the `EXTERNAL` keyword is required and will create a Hive table that references an existing Kudu table. Dropping the external Hive table will not remove the underlying Kudu table.

```
CREATE EXTERNAL TABLE kudu_table (foo INT, bar STRING, baz DOUBLE)
STORED BY 'org.apache.hadoop.hive.kudu.KuduStorageHandler'
TBLPROPERTIES (
  "kudu.table_name"="default.kudu_table",
  "kudu.master_addresses"="localhost:7051"
);
```

In the above statement, normal Hive column name and type pairs are provided as is the case with normal create table statements. The full `KuduStorageHandler` class name is provided to inform Hive that Kudu will back this Hive table. A number of TBLPROPERTIES can be provided to configure the KuduStorageHandler. The most important property is `kudu.table_name` which tells hive which Kudu table it should reference. The other common property is `kudu.master_addresses` which configures the Kudu master addresses for this table. If the `kudu.master_addresses` property is not provided, the `hive.kudu.master.addresses.default` configuration will be used.

Impala Tables

Because Impala creates tables with the same storage handler metadata in the HiveMetastore, tables created or altered via Impala DDL can be accessed from Hive. This is especially useful until HIVE-22021 is complete and full DDL support is available through Hive. See the Kudu documentation and the Impala documentation for more details.

Data Ingest

Though it is a common practice to ingest the data into Kudu tables via tools like Apache NiFi or Apache Spark and query the data via Hive, data can also be inserted to the Kudu tables via Hive INSERT statements. It is important to note that when data is inserted a Kudu UPSERT operation is actually used to avoid primary key constraint issues. Making this more flexible is tracked via HIVE-22024. Additionally UPDATE and DELETE operations are not supported. Enabling that functionality is tracked via HIVE-22027.

Examples

```
INSERT INTO kudu_table SELECT * FROM other_table;

INSERT INTO TABLE kudu_table
VALUES (1, 'test 1', 1.1), (2, 'test 2', 2.2);
```