

Purpose

The aim of the project is to reroute the queries that can't be answered directly by Kylin to Hive on Spark.

What Kylin currently does

- It uses Hive to build cubes.
- The queries that can't be directly answered using cubes are redirected to Hive.

Problems with the current approach

- It takes huge time for joining tables using Hive.
- Query redirection to Hive in case they can't be directly answered using cubes makes the response time enormous due to Hive latency. Due to such latency this feature is currently turned off.

Our proposal

- We can use Hive on Spark instead of native Hive to leverage Spark benefits. This would make response to the rerouted queries much faster.
- Hive on Spark can also be used for building cubes. (Not under the scope).

Other approaches

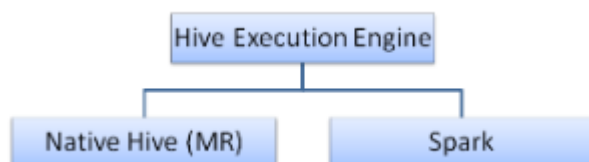
- Instead of sending the unsupported queries to Hive on Spark, we could also use Spark SQL.
- Using Shark is no longer a good option, because its development has ended.

Why is our approach better

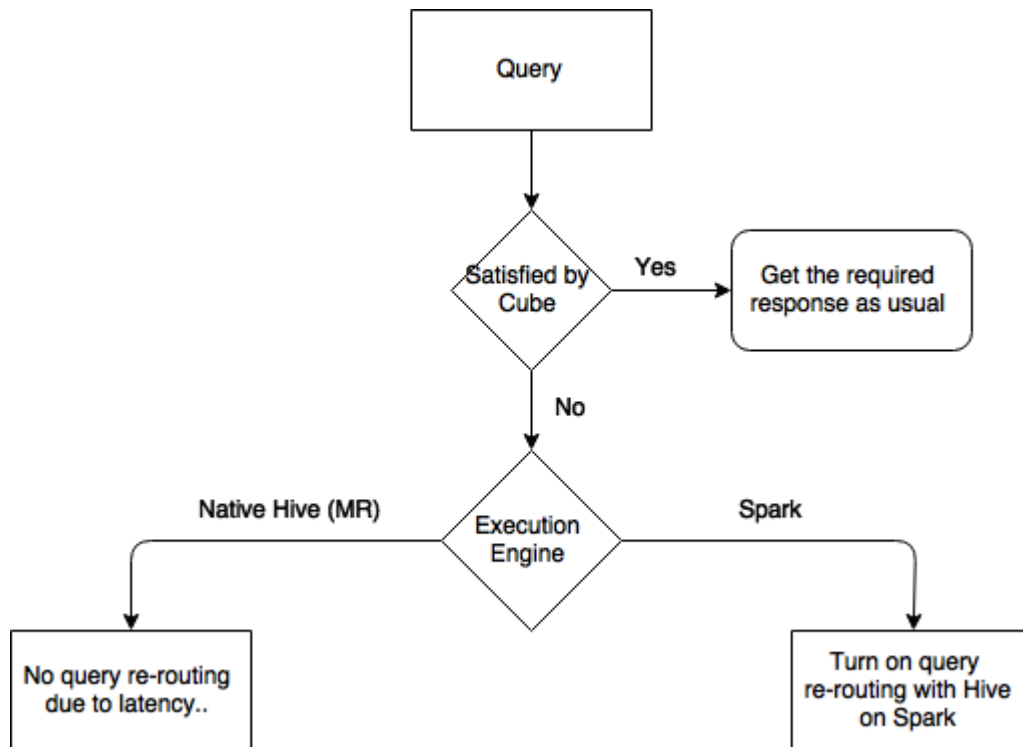
- We have an existing investment in Hive.
- We would like to migrate to Spark.
- The best way would be to add support for Spark as an alternative execution engine to Hive.
- It would be a clear path for us to migrate to Spark.
- As we already have queries formulated for Hive, the execution on Hive on Spark would let us leverage the power of Spark with a minimum code change.
- It would be easier and faster to implement Hive on Spark than other alternatives.

Design

We expect to provide users an option to set Spark as execution engine for Hive, that would enable answering all the Hive supported queries, instead of just the queries that can be answered using cubes.



This is how the query will be handled. In case the query can't be satisfied by the cube, then if Spark as execution engine is enabled then the query will be rerouted to Hive on Spark otherwise the feature will be turned off.



When is a query re-routed?

We need to identify when is a query not answered by a cube. An approach can be to put a try catch in the code when a query is made to the cube. If the cube returns an exception then the query needs to be rerouted to Hive on Spark. We can use this approach to start with.

QueryService.query() would be a good point to capture failure and redirect to Hive on Spark.

Putting a try catch around the query(), the following exceptions would indicate a failure by Kylin, and would make Hive on Spark worth trying:

- org.eigenbase.sql.validate.SqlValidatorException -- as the cause of SQLException - This is thrown when an unknown column or table appears in SQL, unknown means not captured by an cube
- org.eigenbase.sql.parser.impl.ParseException -- as the cause of SQLException - This is thrown when some unknown SQL syntax appears in SQL, e.g. some Hive specific grammar.

After successful execution

If the query is executed successfully on Hive on Spark, then it could be returned back in a SQLResponse object.

This way QueryController would consider it a successful execution and would cache the query and the results.

Miscellaneous

Test Environment

- Hadoop 2.6.0
- Hive 1.2.0
- Spark 1.3.1