Kite Connector Hbase support

Design doc for Kite Connector supporting Hbase for basic reads/ writes and DFM(delta fetch merge) if possible

JIRA: https://issues.apache.org/jira/browse/SQOOP-1744 and its sub-tickets.

Summary

Currently we have KiteConnector in Sqoop2 (as of writing this doc) with support for writing to and reading from a HDFS dataset. The goal of SQOOP-1744 is to extend it to support reading from and writing to Hbase data set as well. An additional goal will be to support reading delta records and writing delta records from/to hbase using the Kite SDK/ APIs.

Background

There is no design or feature doc yet written for the details of the KiteConnector. Here are the relevant JIRA tickets that provide details on how the Kite FROM and Kite TO connectors work.

Kite FROM part : https://issues.apache.org/jira/browse/SQOOP-1647

Kite TO part (for writing to HDFS via Kite) : https://issues.apache.org/jira/browse/SQOOP-1588

UPDATE: A design wiki was added later on Kite Connector Design

Requirements

1. Ability for the user to read from and write to Hbase by choosing the Kite connector, It is implementation detail if we choose to have a standalone Kite-Hbase connector reuse the KiteConnector we have today in some fashion to indicate the data set we will use
2. Ability to indicate the partition strategy and column/counter/key mapping for hbase data sets
3. Ability to support delta reads and writes to the Hbase
4. Integration tests to prove that we can move data from the JDBC to Hbase and vice versa
5. Also if we can make use of Avro IDF it would avoid all the unnecessary back and forth between avro and sqoop object array types to improve the performance.

Design

Overall there are 2 ways to implementing this functionality using the KiteSDK

~~**Option 1**~~

~~Duplicate a lot of the code in KiteConnector and add a new independent connector for KiteHbaseConnector. The major con is the code duplication and effort to support Yet another connector~~

Option 2:

- Use the current KiteConnector and add a enum to select the type of dataset Kite will create underneath, or parse to URI given in the FromJobConfig and ToJobConfig to figure out the dataset to be HIVE/ Hbase or HDFS

```
public enum DataSetType {
  HDFS,
  HBASE,
  HIVE
}
// use this enum to determine what dataset kite needs to create underneath
  @Input
  public DataSetType datasetType

or
// parse this to figure out the data set
  @Input(size = 255, validators = {@Validator(DatasetURIValidator.class)})

  public String uri
```

- Piggy back on config annotations (conditions that we are intending to add since ages!) to show only relevant config subsequently. =
Pros :

1. No code duplication
2. No weird build dependency of KiteHbaseConnector depending on KiteConnector that might make independent connector upgrade complicated

Implementation Details (With Option#2)

- Use uri to parse the type of dataset in the connector
- Rely on the uri for Hbase dataset to be setup with relevant mappings. Rely on Kite-HDFS partitioning for hbase partitioning strategy setup
- KiteExtractor to support creating Hbase datasets via Kite SDK and reading records and piggyback on partitioning implementation of Kite-HDFS
- KiteLoader to support creating Hbase datasets via Kite SDK and writing records. Unlike Kite-HDFS that has the ability to create temp datasets and merge them only when job succeeds (commit phase), in case of Hbase we cannot do that, we have to commit as we write. We are aware that at this point if a job/task failure happens, there can be partial commits and or dupes.
- If we support DFM, add relevant DFM configs and code in KiteConnector - TBD

Testing

Integration test suite will be enhanced to add support for the JDBC-KiteHBaseConenctor and vice versa

Performance Testing

None at this point

Open Questions

1. Can we make the IDF as a config option? so that we can dynamically choose to set the IDF (csv or avro?) Avro IDF has a great performance benefits to the Kite Connector since natively kite store avro records in memory, In that case, the Hbase can use AvroIDF
2. ~~Do we really independent connectors for Kite Hbase, Kite Hive, seems like a overkill to me.~~
3. Do we need to store zookeepr info in linkConfig? like Hdfs port is in linkconfig?

   `hbase:<zookeeper>/<dataset-name>`

   The *zookeeper* argument is a comma separated list of hosts. For example

   `hbase:host1,host2:9999,host3/myDataset`