Hive across Multiple Data Centers (Physical Clusters)

This project has been abandoned. We're leaving the design doc here in case someone decides to attempt this project in the future.

Use Cases

Inside facebook, we are running out of power inside a data center (physical cluster), and we have a need to have a bigger cluster. We can divide the cluster into multiple clusters - multiple hive instances, multiple mr and multiple dfs. This will put a burden on the user - he needs to know which cluster to use. It will be very difficult to support joins across tables in different clusters, and will lead to a lot of duplication of data in the long run. To get around these problems, we are planning to extend hive to span multiple data centers, and make the existence of multiple clusters transparent to the end users in the long run. Note that, even today, different partitions/tables can span multiple dfs's, and hive does not enforce any restrictions. Those dfs's can be in different data centers also. However, a single table/partition can only have a single location. We need to enhance this. We will not be able to partition our warehouse cluster into multiple disjoint clusters, and therefore some tables/partitions will be present in multiple clusters.

Requirements

In order to do so, we need to make some changes in hive, and this document primarily talks about those. The changes should be generic enough, so that they can be used by others (outside Facebook) also, if they have such a requirement. The following restrictions have been imposed to simplify the problem:

- There will be a single hive instance, possibly spanning multiple clusters (both dfs and mr)
- There will be a single hive metastore to keep track of the table/partition locations across different clusters.
- A table/partition can exist in more than one cluster. A table will have a single primary cluster, and can have multiple secondary clusters.
- Table/Partition's metadata will be enhanced to support multiple clusters/locations of the table.
- All the data for a table is available in the primary cluster, but a subset can be available in the secondary cluster. However, an object (unpartitioned table/partition) is either fully present or not present at all in the secondary cluster. It is not possible to have partial data of a partition in the secondary cluster.
- The user can only update the table (or its partition) in the primary cluster.
- The following mapping will be added. Cluster -> JobTracker
- By default, the user will not specify any cluster for the session, and the behavior will be as follows:
    - The query will be processed in a single cluster, and use the jobtracker for that cluster.
    - If the primary cluster of any output table is different from the query processing cluster, an error is thrown. So, a multi-table insert with tables belonging to different primary clusters will always fail.
    - If the input's table primary cluster is different from the query processing cluster, the query will only succeed if all the partitions for that input table are also present on the query processing cluster.
    - If an output is specified, the primary cluster for that output will be used.
    - If the output specified is a new table, the output is not used in determining the query processing cluster.
    - If no output is specified (or the output is a new table), and there are multiple inputs for the query, all the input tables primary clusters are tried one-by-one, till a valid cluster is found.

- Few examples will illustrate the scenario better:
- Say T11, T12, T21, T31 are tables belonging to cluster C1, C1, C2 and C3 respectively, and it has no secondary clusters.
    - The query 'select .. from T11 .. ' will be processed in C1
    - The query 'select .. from T11 join T12 .. ' will be processed in C1
    - The query 'select .. from T21 .. ' will be processed in C2
    - The query 'select .. from T11 join T21 .. ' will fail
    - 'Insert .. T13 select .. from T11 ..' will be processed in C1 and the T13 will be created in C1
    - 'Insert .. T21 select .. from T11 ..' will fail

- If we change the example slightly:
- Say T11, T12, T21, T31 are tables belonging to cluster C1, C1, C2 and C3 respectively. T11's secondary cluster is C2 (and all the data for T11 is also present in C2).
    - The query 'select .. from T11 .. ' will be processed in C1
    - The query 'select .. from T11 join T12 .. ' will be processed in C1
    - The query 'select .. from T21 .. ' will be processed in C2
    - The query 'select .. from T11 join T21 .. ' will be processed in C2
    - The query 'select .. from T11 join T31 .. ' will fail
    - 'Insert .. T13 select .. from T11 ..' will be processed in C1 and the T13 will be created in C1
    - 'Insert .. T21 select .. from T11 ..' will be processed in C2, and T21 will remain in C2

The same idea can be extended for partitioned tables.

- The user can also decide to run in a particular cluster.
    - Use cluster <ClusterName>
- The system will not make an attempt to choose the cluster for the user, but only try to figure out if the query can be run in the <clusterName>. If the query can run in this cluster, it will succeed. Otherwise, it will fail.
- The user can go back to the behavior to use the default cluster.
    - Use cluster

- Eventually, hive will provide some utilities to copy a table/partition from the primary cluster to the secondary clusters.
  In the first cut, the user needs to do this operation outside hive (one simple way to do so, is distcp the partition from the primary to the secondary cluster, and then update the metadata directly - via the thrift api).

- This will require a change to the metastore schema. StorageDescriptor will be enhanced to add:
  PrimaryCluster - ClusterStorageDescriptor
  and SecondaryClusters - Set<ClusterStorageDescriptor>

The ClusterStorageDescriptor contains the following:
ClusterName
Location

location will be removed from the StorageDescriptor.

- This will require a scheme change and data migration.
- The thrift structure will be backward compatible
  - New entries will be added: ClusterName, IsPrimary etc., but existing clients using sd.location will continue to work

In order to support the above, hive metastore needs to be enhanced to have the concept of a cluster. The following parameters will be added:

- hive.default.cluster.name - For migration, all tables/partitions belong to this cluster.
  All new tables belong to this cluster.
- hive.use.default.cluster - The default cluster will be used depending on input table's schema.

The existing thrift API's will continue to work as if the user is trying to access the default cluster.
New APIs will be added which take the cluster as a new parameter. Almost all the existing APIs will be enhanced to support this. The behavior will be the same as if, the user issued the command 'USE CLUSTER <CLUSTERNAME>

- A new parameter will be added to keep the filesystem and jobtrackers for a cluster
  - hive.cluster.properties: This will be json - ClusterName -> <FileSystem, JobTracker>
  - use cluster <cluster name> will fail if <cluster name> is not present hive.cluster.properties
  - The other option was to support create cluster <> etc. but that would have required storing the cluster information in the metastore including jobtracker etc. which would be difficult to change per session.

If the user does not intend of use multiple clusters, there should be no change in the behavior of hive commands, and all the existing thrift APIs should continue to work.

An upgrade script will be provided to upgrade the metastore with the patch.