ListBucketing

- Goal
  - Basic Partitioning
  - List Bucketing
  - Skewed Table vs. List Bucketing Table
  - List Bucketing Validation
    - DDL
    - DML
    - Alter Table Concatenate
  - Hive Enhancements
    - Create Table
    - Alter Table
      - Alter Table Skewed
      - Alter Table Not Skewed
      - Alter Table Not Stored as Directories
      - Alter Table Set Skewed Location
    - Design
- Implementation

Goal

The top level problem is as follows:

There are many tables of the following format:

- `create table T(a, b, c, ....., x) partitioned by (ds);`

and the following queries need to be performed efficiently:

- `select ... from T where x = 10;`

The cardinality of 'x' is in 1000's per partition of T. Moreover, there is a skew for the values of 'x'. In general, there are ~10 values of 'x' which have a very large skew, and the remaining values of 'x' have a small cardinality. Also, note that this mapping (values of 'x' with a high cardinality) can change daily.

The above requirement can be solved in the following ways:

Basic Partitioning

Create a partition per value of 'x'.

- `create table T(a,b,c, .......) partitioned by (ds, x);`
- Advantages
  - Existing Hive is good enough.
- Disadvantages
  - HDFS scalability: Number of files in HDFS increases.
  - HDFS scalability: Number of intermediate files in HDFS increases. For example if there are 1000 mappers and 1000 partitions, and each mapper gets at least 1 row for each key, we will end up creating 1 million intermediate files.
  - Metastore scalability: Will the metastore scale with the number of partitions.

List Bucketing

The basic idea here is as follows: Identify the keys with a high skew. Have one directory per skewed key, and the remaining keys go into a separate directory. This mapping is maintained in the metastore at a table or partition level, and is used by the Hive compiler to do input pruning. The list of skewed keys is stored at the table level. (Note that initially this list can be supplied by the client periodically, and eventually it can be updated when a new partition is being loaded.)

For example, the table maintains the list of skewed keys for 'x': 6, 20, 30, 40. When a new partition is being loaded, it will create 5 directories (4 directories for skewed keys + 1 default directory for all the remaining keys). The table/partition that got loaded will have the following mapping: 6,20,30,40,others. This is similar to hash bucketing currently, where the bucket number determines the file number. Since the skewed keys need not be consecutive, the entire list of skewed keys need be stored in each table/partition.

When a query of the form

- `select ... from T where ds = '2012-04-15' and x = 30;`

is issued, the Hive compiler will only use the directory corresponding to x=30 for the map-reduce job.

For a query of the form

- `select ... from T where ds = '2012-04-15' and x = 50;`

the Hive compiler will only use the file corresponding to x=others for the map-reduce job.

This approach is good under the following assumptions:

- Each partition's skewed keys account for a significant percentage of the total data. In the above example, if the skewed keys (6,20,30 and 40) only occupy a small percentage of the data (say 20%), the queries of the form x=50 will still need to scan the remaining data (~80%).
- The number of skewed keys per partition is fairly small. This list is stored in the metastore, so it does not make sense to store 1 million skewed keys per partition in the metastore.

This approach can be extended to the scenario when there are more than one clustered key. Say we want to optimize the queries of the form

- `select ... from T where x = 10 and y = 'b';`
- Extend the above approach. For each skewed value of (x,y), store the file offset. So, the metastore will have the mapping like: (10, 'a') -> 1, (10, 'b') -> 2, (20, 'c') -> 3, (others) -> 4.

  A query with all the clustering keys specified can be optimized easily. However, queries with some of the clustering keys specified:

  - `select ... from T where x = 10;`
  - `select ... from T where y = 'b';`

  can only be used to prune very few directories. It does not really matter if the prefix of the clustering keys is specified or not. For example for x=10, the Hive compiler can prune the file corresponding to (20, 'c'). And for y='b', the files corresponding to (10, 'a') and (20, 'c') can be pruned. Hashing for others does not really help, when the complete key is not specified.

This approach does not scale in the following scenarios:

- The number of skewed keys is very large. This creates a problem for metastore scalability.
- In most of the cases, the number of clustered keys is more than one, and in the query, all the clustered keys are not specified.

Skewed Table vs. List Bucketing Table

- *Skewed Table* is a table which has skewed information.
- *List Bucketing Table* is a skewed table. In addition, it tells Hive to use the list bucketing feature on the skewed table: create sub-directories for skewed values.

A normal skewed table can be used for skewed join, etc. (See the Skewed Join Optimization design document.) You don't need to define it as a list bucketing table if you don't use the list bucketing feature.

List Bucketing Validation

Mainly due to its sub-directory nature, list bucketing can't coexist with some features.

DDL

Compilation error will be thrown if list bucketing table coexists with

- normal bucketing (clustered by, tablesample, etc.)
- external table
- `"load data ..."`
- CTAS (Create Table As Select) queries

DML

Compilation error will be thrown if list bucketing table coexists with

- `"insert into"`
- normal bucketing (clustered by, tablesample, etc.)
- external table
- non-RCfile due to merge
- non-partitioned table

Partitioning value should not be the same as a default list bucketing directory name.

Alter Table Concatenate

Compilation error will be thrown if list bucketing table coexists with

- non-RCfile
- external table for alter table

Hive Enhancements

Hive needs to be extended to support the following:

Create Table

```
CREATE TABLE <T> (SCHEMA) SKEWED BY (keys) ON ('c1', 'c2') [STORED AS DIRECTORIES];
```

The table will be a skewed table. Skewed information will be created for all partitions.

For example:

- `create table T (c1 string, c2 string) skewed by (c1) on ('x1') stored as directories;`
- `create table T (c1 string, c2 string, c3 string) skewed by (c1, c2) on (('x1', 'x2'), ('y1', 'y2')) stored as directories;`

'STORED AS DIRECTORIES' is an optional parameter. It tells Hive that it is not only a skewed table but also the list bucketing feature should apply: create sub-directories for skewed values.

Alter Table

Alter Table Skewed

```
ALTER TABLE <T> (SCHEMA) SKEWED BY  (keys) ON ('c1', 'c2') [STORED AS DIRECTORIES];
```

The above is supported in table level only and not partition level.

It will

- convert a table from a non-skewed table to a skewed table, or else
- alter a skewed table's skewed column names and/or skewed values.

It will impact

- partitions created after the alter statement
- but not partitions created before the alter statement.

Alter Table Not Skewed

```
ALTER TABLE <T> (SCHEMA) NOT SKEWED;
```

The above will

- turn off the "skewed" feature from a table
- make a table non-skewed
- turn off the "list bucketing" feature since a list bucketing table is a skewed table also.

It will impact

- partitions created after the alter statement
- but not partitions created before the alter statement.

Alter Table Not Stored as Directories

```
ALTER TABLE <T> (SCHEMA) NOT STORED AS DIRECTORIES;
```

The above will

- turn off "list bucketing"
- not turn off the "skewed" feature from table since a "skewed" table can be a normal "skewed" table without list bucketing.

Alter Table Set Skewed Location

```
ALTER TABLE <T> (SCHEMA) SET SKEWED LOCATION (key1="loc1", key2="loc2");
```

The above will change the list bucketing location map.

Design

When such a table is being loaded, it would be good to create a sub-directory per skewed key. The infrastructure similar to dynamic partitions can be used.

`Alter table <T> partition <P> concatenate;` needs to be changed to merge files per directory.

Implementation

> **Version information**
>
> List bucketing was added in Hive 0.10.0 and 0.11.0.

HIVE-3026 is the root JIRA ticket for the list bucketing feature. It has links to additional JIRA tickets which implement list bucketing in Hive, including:

- HIVE-3554: Hive List Bucketing – Query logic (release 0.10.0)
- HIVE-3649: Hive List Bucketing - enhance DDL to specify list bucketing table (release 0.10.0)
- HIVE-3072: Hive List Bucketing - DDL support (release 0.10.0)
- HIVE-3073: Hive List Bucketing - DML support (release 0.11.0)

For more information, see Skewed Tables in the DDL document.