Snapshot Restore

Given the “take a snapshot” functionality you want the capability to:

- Rollback the current table to a previous state.
- Look at the current table side-by-side with a specified snapshot (before making the decision whether to rollback or not)
- Been able to pick “individual items” from a specific snapshot (only some small range of data was “lost” from the current table)

What is a Snapshot?

A snapshot is the dump of all the “metadata” information needed to recreate a table.

- The Table Descriptor (column families, and other table info).
- All the Table’s Regions Info (start key, end key for each region, …).
- A Reference to each HLog.
- A Reference to each HFile.

Since the reference to the HLogs and HFiles is weak, some more logic is needed to keep around HFiles after a compaction or region deletion, or HLogs after deletion. So instead of deleting the files we can just move them to an “archive” folder.

At this point our “reference” is even more weak and we need something like:

- If the Referenced file doesn’t exists in the expected path
    - Try to lookup the file from the archive (assert “must be there”)

What is a Restore?

Given the initial 3 goals, the “Snapshot Restore” is the capability to create/populate a table given
all the dumped “metadata” information of another table (aka snapshot).

The difference between the first point and the other two is that when you do a rollback you’ve data in the table, and some data can be the same as the snapshots (same hfiles).

I’ll try to describe the two operations as:

- Restore: Given a empty or non existent table, populate it with the snapshot content.
- Rollback: Replace the current table content with an older version of the data.

How Restore Works

Since the table is empty or non-existent we’re in a situation where the master can do all the work. If the table exists we’ve to check if the table matches the snapshot table descriptor otherwise we need to create the table from the snapshot descriptor.

Once we have the new table, for each region listed in the snapshot:

- Create a new region in the new table with the same start/end key (and other info...)
- Create a “Reference” to the hfiles in the original table.
- Copy the HLog referenced by the snapshot.
- Add the region to .META.

How Rollback Works

The rollback is a little bit tricky since the table is online.
Before starting the restore The Master needs to check what are the differences between the current table and the snapshot.

If the table is completely different from the snapshot (there's no intersection between the regions in the current table and the regions in the snapshot, due to split or similar operations) we can "disable" the table and apply the Restore operations.

If the table is quite similar (has intersections between the regions in the snapshot) we need to:

- Remove the regions that are not part of the snapshot
- Notify each Region Server to use the snapshot version of the hfiles.
- Add the regions that are part of the snapshot but not of the current table.

In this way we can do a rollback of the table without having to disable the table.

What is a "Reference"?

In the "Restore" I've mentioned that the HMaster create a "Reference" the hfiles in the original table. Since HFiles are immutable two tables can share one or more files.

The only problem that we have is related with the current HBASE.ROOT layout or with the missing Hardlink functionality in HDFS (See HBASE-6233 as reference).
And basically the problem is that the hfiles are stored inside a /table/region/cf/ path. This means that each Table/Region is responsible to manage its own files, and means that on compaction or region deletion this files are removed.

But, as we've described in the beginning: instead of deleting the files we can just move them to an "archive" folder.

At this point we can introduce a "HFile Link" that is a special kind of "Reference" that HBase see as a "copy" of the original file. The underlying implementation is a special stream that if the referred hfile exists in the original table, it reads from that location, otherwise reads from the archive folder. This is done by catching the exceptions thrown from the reads and other stream operations and switching to the archived version.

In this way, when a table is restored we don't need to copy Terabytes of data, and we don't need to care about when the original table "delete" the hfile due to a compaction/region deletion, since the file is moved in the "archive" folder and the "HFile Link" switch automatically "its link" to the archive directory.