HBase Backup and Restore (HBase-7912)

V1: March 31, 2014

V2(Last updated): June 4, 2014

Introduction

HBase Backup and restore is a crucial requirement by enterprises using HBase as data repository. Some solutions have been attempted in the past, for example, solutions using HBase Export and Import API. Given the availability of some important new HBase features, notably HBase snapshot (HBASE-6055, HBASE-7290), we propose a new HBase Backup and Restore design based on these new features. The benefit is ease of use and better performance and accuracy.

Backup and restore is used for data recovery in case of data loss or failures. HBase snapshot enables the efficient and effective capture of a consistent and point-in-time (both to some degree) HBase table image. But snapshot alone is not enough for a complete backup and restore/recovery solution. The snapshot information and data is tightly coupled and stored with the existing HBase cluster -- in-place backup. We want to backup HBase data to FileSystem cross clusters and possible other storage media or servers. This proposal is a logical extension to the snapshot feature.

2. Key features and Use Cases

A common practice of backup and restore in database is to first take full baseline backup, and then periodically take incremental backup that capture the changes since the full baseline backup. HBase cluster can store massive amount data. Therefore we want use full backup in combination with incremental backups for HBase as well. The following is a typical use case scenario for full and incremental backup.

a. The user takes a full backup of a table or a set of tables in HBase.
b. The user schedules periodical incremental backups to capture the changes from the full backup, or from last incremental backup.
c. The user needs to restore table data to a past point in time.
d. The full backup is restored to the table(s) or to different table name(s). Then the incremental backups that are up to the desired point in time are applied on top of the full backup.

We would support the following key features and capabilities.

- Backup to FileSystem cross clusters and possible other storage media or servers.
- Full backup uses HBase snapshot to capture HFiles.
- We use HBase WALs to capture incremental changes, but we use bulk load of HFiles for fast incremental restore.
- Support single table or a set of tables backup and restore.
- Restore to different table names.
- Support adding and removing tables to and from backup set without interruption of incremental backup schedule.
- Support merge of incremental backups into longer period and bigger incremental backups for easy storage and restore.
  Unified command line interface for all the above

To illustrate these key capabilities, the following are two more detailed use case examples.

Use case example 1:

a. The user takes a full backup of a set of tables (i.e. table1 and table2) in HBase.
b. The user takes incremental backups. The incremental backup will only track table1 and table2.
c. The user adds other tables (i.e. table 3 and table4) in Hbase, and an implicit full backup is executed during the add process
d. The user continues to take incremental backups. The incremental backup data would cover table1, table2, table3 and table4.
e. The user wants to restore table3 and table4 to a past PIT.
f. The full backup in c. is restored onto HBase cluster. Then the incremental backups after that full backup are applied on top of the full restore until the PIT.

Use case example 2:

a. The user takes a full backup of a set of tables in HBase.
b. The user takes daily incremental backups.
c. The user merges the daily incremental backups into weekly incremental backups.
d. The user combines/rolls up the weekly incremental backup into monthly incremental backups.
e. The user wants to restore the tables to a past PIT.
f. The full backup is restored onto HBase cluster.
g. The monthly incremental backups before the desired PIT are applied.
h. The closest daily backups up to the PIT are applied.

3. Overall Design

1) Design Summary

The solution covers the two main parts of backup and restore.

a. Full backup and restore. Backup will first invoke HBase snapshot and export snapshot internally. The full backup can be restored with HBase bulk import utility.

b. Incremental backup uses WALs to capture the data changes since last full backup or incremental backup. We execute roll log across region servers to track the WALs that need to be in the backup. Then a distributed copy is used to move the physical files to target FileSystem. The WALs can be replayed into HFiles on the fly or offline. During restore, the HFiles will be applied on top of the full backup via HBase Bulk Load utility.

2) Client Side

The client initiates request to start a full backup or incremental backup via a command line interface. The typical input of this request is

a. Type (full or incremental)

b. The table names.
c. The directory location where the backed-up files are to be copied.
d. Other options.

The client synchronously waits for the completion (either successful or failed) of this request.

3) Backup Manager

The Backup client passes the backup request to the Backup Manager. Backup manager checks the type of the backup request and delegate accordingly, and act as a singleton to make sure only one backup is on-going. Backup Manager can also do other book-keeping work or cleanup work as needed.

The Backup Manager will pass the request to Backup Handler for execution of the backup.

4) Tracking Progress and Aborting

The user should be able to track the progress of the backup process. The user should be able to abort the process if the user chooses to do so, for example, in the case of a performance drag. The user should be able to track the restore progress as well and abort restore progress when necessary.

5) Backup Manifest and history

We want to have a manifest file for each backup. The manifest file will have such information as the type of the backup, the size, the location info, etc. The manifest will also provide the dependency lineage info needed to restore the backup. For example, an incremental restore will need to know the required full backup or incremental backups that are needed before it can be applied.

6) Adding and removing table to backup set

The user can add or remove a table (or tables) to or from backup set at a point in time.

a) backup add <table-name>
b) backup remove <table-name>

When a table (or tables) is added to backup set, we will do an initial full backup for the table(s), and any incremental backup after that point in time will include the table(s).
When a table (or tables) is removed from backup set, any incremental backup after that point in time will filter out the table(s) if needed
Adding or removing a table to or from backup set will not alter incremental backup sequence for existing tables tracked for incremental backup that are not part of the add/remove. That means later increment backup will not be from this add/remove action to the current for these tables. Later incremental backup will still from last incremental backup to current.

4. Full Backup

1) Take snapshot

We generate a snapshot request by using a snapshot name suffixed with the current timestamp. For example, 'backup_ 1360652124199'. We then invoke the Snapshot Manager to take the snapshot and waits for its completion status.

2) Existing Snapshot

We can allow the use of an existing snapshot by allowing the user to give a snapshot name. In this case, no new snapshot will be taken. The existing snapshot will be verified to exist and then the data will be copied to the target backup location.

3) From full backup to incremental backup

We use WALs to track incremental changes. An important step is a clean cut on what need to be included in the next incremental backup after a full backup. We will do a log roll on each RS, and each RS will record its current WAL number. The information will be saved for later incremental backup. This current WAL will be included in the next incremental backup files, but only the content after the recorded WALnumber will be used.

4) Export Snapshot

We will invoke HBaseAdmin export snapshot call to copy out the full backup data.

5) Full restore

The backed-up HFiles will be restored using HBase bulk load utility. Tables can be created if not exist.

5. Incremental Backup

1) Steps

When the client issues an incremental backup request, BackupManager will check the backup request and then kicks off a global procedure via HBaseAdmin for all the active region servers to roll log. Each region server will again record their log number onto Zookeeper. We will determine what log files need to be include in this incremental backup, and use DistCp to copy them to target location.

2) Converting Incremental HLogs into HFiles and Incremental Restore

We'll convert/replay the backed-up Hlogs into HFiles for fast incremental restore. This will done offline without impacting the running HBase instance. When we relay the HLogs, we will only filter to include the tables that we need to include. Therefore only the HFiles that cover the backed-up tables are stored as incremental backup.

3) Merging Backups

After converting incremental HLogs to HFiles, we will have a tool to merge incremental backups or/and full backups. The HFiles from incremental backup will be combined to form a bigger longer period incremental backup image. e.g. daily incremental backup can be combined to form weekly incremental backup. Incremental backups can also be merged with their parent full backup.

The incremental backup images after convert and merge will be restored to a cluster via HBase bulk load.

6. Other consideration and limitations

- **Abort Backup :** If there is a failure(such as RegionServer crash), the backup process will abort itself, and clean up all the logs and zookeeper states. We are considering to keep a mid-state so the next time the process won't start from beginning. This is a future improvement item
- **Life time of Logs**: Since incremental backup heavily relies on Hlogs to capture the delta between each backup. A BackupLogCleaner class is extended from BaseLogCleanerDelegate, as part of hbase.master.logcleaner.plugins. It would keep the logs from archived/deleted until an incremental backup occur and free them up.
- **BulkLoad and WritetoWal(false):** Unfortunately incremental backup can't support either scenario since no Log will be written. User will have to do a full backup or export data manually.
- **Single Backup Process**: currently the logic support only one backup process at a time, and plan to be relaxed in the future.