HBase Snapshot

Li Chongxin

1 Introduction

Snapshot of HBase[1] takes a copy of the specified table at a particular point in time. Although Hadoop has provided backup to protect from failed servers, this does not protect use from software bugs that might delete or alter data in ways we did not plan. We should have a way that we can roll back a dataset. The snapshot of the table can be used as a backup of the current table.

2 Requirements

Requirements for snapshot of HBase:

1) Only the HBase admin can create a snapshot.
2) Snapshot can be created for both online (enabled) tables and offline (disabled) tables.
3) Snapshot is the copy of a specified table at a particular point, thus updates that happen at the time before this point, no matter whether the data have been flushed onto the disk, should have a reflection in the snapshot. On the contrary, updates that happen later than this point should not be included in the snapshot.
4) Snapshot of a table should not disable the table itself. That is during the creation of a snapshot, the table could continue to take on reads and writes.
5) Latency of creating a snapshot should be small. Splits and compactions as well as memstore flushing might be suspended during snapshot. An OOME might occur if the time to create a snapshot is too long.
6) Only one snapshot is permitted at a time. If one snapshot is ongoing, other snapshot requests should be disallowed.
7) There should be a mechanism to restore a table from a snapshot.
8) There should be a mechanism to list all the snapshots of a given table, including snapshot name and creation time.
9) There should be a mechanism to delete a snapshot.
10) A table could have multiple snapshots at different time. Deletion of one snapshot or even the table itself should not impact any other snapshots of the table.
11) There should be a mechanism to export and import the snapshot. The exported format should be the same as how a table is exported so that the exported snapshot can be imported to another data center as an exported table.

[1] https://issues.apache.org/jira/browse/HBASE-50

3 Design Overview

Snapshot of a table can be divided into three parts:

Message Passing: Tables of HBase can be enabled and disabled. For table that is disabled (offline), it is not served by any region server. So we don't have to pass any message to the region servers. The snapshot can be created directly by the master.

But for tables that are enabled (online), snapshot request must be spread over the cluster to trigger the creation of snapshot. This can be done by ZooKeeper. However, several conditions must be met before the real action starts, such as the table is in good status and all the regions of the given table must be online. All these information are transmitted between region servers and master via ZooKeeper. Once the conditions are guaranteed, snapshot can be started on region servers which hold the table regions.

Snapshot Creation: Data of an HBase table is mainly comprised of the content under */hbase/<tablename>* directory in HDFS. For online tables, there also would be data in the memory (memstore). All these data should be included when the snapshot is created.

Under directory */hbase/<tablename>*, the amount of data can be very huge for an HBase table, so we can't simply copy the table files to create a snapshot for the consideration of latency. Since HBase files are all immutable once created, we can just take references of the data files (HFile) to create the snapshot instead of copying actual data files. It is much faster to create references than make a copy of data. At the same time, metadata should also be dumped. That's all what we do to create a snapshot for an offline table.

However, for tables that are online, there are still data in the memstore. Because cache flushing is time consuming, to make snapshot window narrow, we just roll the region server logger and dump a file that lists the names of current logs. Data in the memstore can be reconstructed from the logs when restore.

Snapshot Maintenance: In the above step, snapshot is made up of references to data files and a list of file names for the logs. Thus files (including HFiles and log files) that are used by snapshots should not be deleted or mutated as before.

In current implementation, HBase data files can be deleted in several scenarios, such as compaction, split and table deletion. To support snapshot, these functions will be modified so that old files will not be deleted but archived to a shadow directory if it is referred by a snapshot. And files in this directory can only be deleted when there is no reference to this file.

Things work the same for the log files. But the difference is that old log files have already been archived in a directory. An old log file is cleaned from the archive dir if it is no longer used by other functions any more. With snapshot, we just further check if a log file is used by a snapshot and then decide whether to delete it or not.

The implementation details are described in the following three chapters.

4 Message Passing

Snapshot starts at **HBaseAdmin**, where table status is verified first. If a table is offline, that is all its regions are offline, the snapshot request will be only sent to the master as other requests. Because the table is not served by any region server, the master will create the snapshot completely. On the other side, if a table is online, that is all its regions are online (parent region after split will not be included), the snapshot request should be sent

to all region servers. And snapshot is created by region servers that serve the table regions. Tables that are partially open or closed are not allowed to create a snapshot.

Although clients communicate with each region server through RPC currently, there is not a way for a client to RPC all members of the cluster. ZooKeeper is used to spread the snapshot request over the cluster. To start snapshot via ZooKeeper, all region servers would watch a "*snapshot*" znode. Then the client would trigger the snapshot by touching this znode. The region server watchers would be notified. Each region server will also check its status and a znode will be created under "*ready*" if the region server is ready for snapshot. When all region servers are ready, snapshot can be started on each region servers. This method guarantees that if any one region server is not ready, snapshot will not be started and the other region servers which are ready will be timeout. (This is a synchronous method, should we consider an asynchronous approach?)

For the region servers which don't serve any region of the specified table, they simply create a znode under "*finish*". While for the region servers which hold the table's regions, they'd do work and also create the finish znode when done. When all have finished, client would report snapshot done. Since master is not involved in snapshot of online table, the client will orchestrate the whole process. Detailed work flow is depicted in Appendix A. and the ZooKeeper znodes hierarchy is illustrated in Fig. 1.

All the znodes created under "*ready*" and "*finish*" should be *EPHEMERAL*. That is znodes will be deleted upon region server's disconnect. If a region server dies during the snapshot (between "*ready*" and "*finish*"), znodes created by this region server will be deleted from ZooKeeper automatically and the client will notice this and then abandon the snapshot.

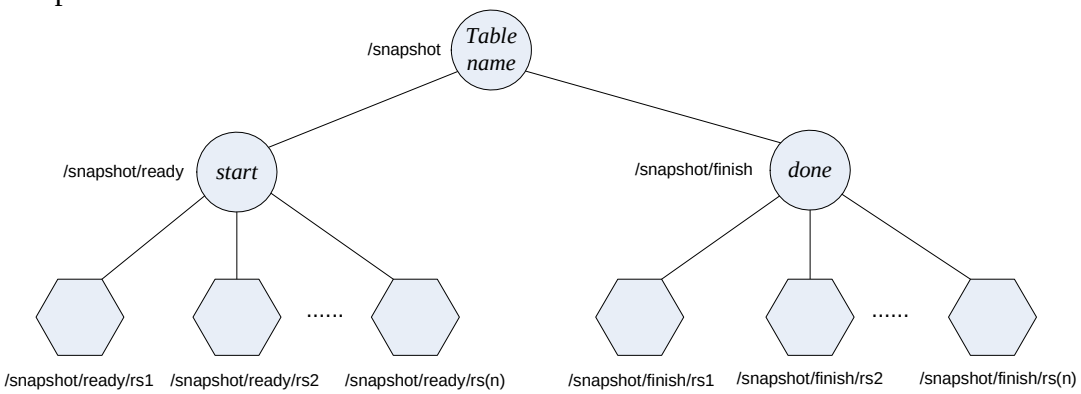


Figure 1 Snapshot Znodes Hierarchy

5 Snapshot Creation

The current state of a table is comprised of the memstore content of all currently running region servers and the content of the */hbase/<tablename>* directory in HDFS. There is also metadata – mainly the current state of its schema and region information – kept up in .META. catalog table. To create a snapshot of a table, all these information should be backed up into the directory */hbase/.snapshot/*, where each snapshot is further kept under a sub-directory <*snapshotname*>. The following sections describe the handling of these data separately.

5.1 Metadata

Metadata of an HBase table is made up of metadata of all regions, including table descriptor, region name as well as start key and end key. For each region of the table, its metadata is saved at two places, .META. table and "*.regioninfo*" file under the region directory in HDFS. However, "*.regioninfo*" is only updated when the region is open and initialized while .META. table is updated each time the region is changed (right?). Therefore, to get the latest metadata of the table, just dump the metadata for each region of the table from .META. and then save it to a file with the same name "*.regioninfo*" under the snapshot directory of the region. Since the metadata for a region is only of KB, this method should be finished very fast. The full path of the backed up metadata will be:

/hbase/.snapshot/<snapshotname>/<encoded-regionname>/.regioninfo

5.2 MemStore Data

As a result of immutability of HDFS files, all the updates (Put, Delete) of an HBase table go to the memstore first, and when the size of memstore is big enough, it is then flushed into an HFile to become persistent in disk. Therefore, when the snapshot is started on a region server for an online table, there still might be un-flushed data for the given table. To back up these memstore data, the straight forward idea is to trigger a cache flush for the table regions. However, cache flushing is time consuming and the thread may be blocked for some time. For the consideration of latency, an alternative method to back up memstore data is to reconstruct the data from logs. And during the creation of snapshot, cache flushing for the regions of this table should be suspended.

When the snapshot is started, the region server first rolls the log writer so that updates before the snapshot are separated from the updates after the snapshot. Given that old log files are now archived instead of deleted, we do not have to copy the log files but just create a file "*.oldlogs*" that lists the names of the log files. The full path of the "*.oldlogs*" is:

/hbase/.snapshot/<snapshotname>/.oldlogs

This operation is performed only once for each region server because there is only one logger for a region server and updates to all the regions are logged into the same log file.

However, log splitting is not carried out here because it also might be take a long time. Data in the old log files is not reconstructed until the snapshot is actually used, such as export and restore in chapter 7.

5.3 HFile Data

Most data of an HBase table are stored as HFile under the directory of */hbase/<tablename>* in HDFS. The data amount of a table can be very huge in HBase, thus it is not appropriate to directly copy the data files to back up the table. As we all know, files in HDFS are immutable once created, to utilize this property, we can just back up the data files by references. For each HFile of a table region, a file with the same name is created under the snapshot directory. Instead of actual data, this file only contains a reference to the origin HFile, just like the reference file after split. But the difference is that reference file after split only contains half of the old file, while the reference file here contains a reference to the entire file (This also speeds up the restoring, see section 7.1). Creating a reference file is much faster that copying the actual data. The full path of a reference file is:

/hbase/.snapshot /<snapshotname>/<encoded-regionname>/<columnfamily>/<filename>

At the same time, reference information for this region must be kept for later use (Chapter 6). A new column family "*snapshot*" is created for .META. table. The qualifier is the HFile name and the value is the count of reference files that refer this HFile. Each time a reference file is created, the column value for this HFile is increased by one. In contrast, if a reference file is deleted by deleting a snapshot, this column value will be decreased by one. This same column family "*snapshot*" will also be applied to -ROOT- table (which keeps the metadata for .META. table) so that snapshot can also be created for .META.

There is one exception for the creation of reference file. If the origin HFile has already been a reference file (after split or restored from snapshot), it should be copied directly instead of creating another reference file to it. And the "*snapshot*" metadata for this region should also be updated.

There is still one last thing to do before the snapshot is finished. Because snapshot information, such as table name and creation time of the snapshot, might be useful for other operations, a Meta info file "*.snapshotinfo*" for the snapshot will be dumped under the directory <*snapshotname*> at the last step of snapshot. The full path of "*.snapshotinfo*" is:

/hbase/.snapshot/<snapshotname>/.snapshotinfo

Therefore, after the above steps (for offline tables, we don't have to perform 5.2), snapshot for both user tables and .META. table can be created under the directory *.snapshot*.

6 Snapshot Maintenance

In the above chapters, snapshot is finally comprised of metadata, a file that lists log files and reference files to the origin HFiles. Although HDFS files for an HBase table are immutable once created, they can still be deleted when compaction and split happens. Deleting the table will also delete all the table files as well. Therefore, special care should be taken if the log files and HFiles are used by snapshots.

The first three sections in this chapter discuss the snapshot maintenance for compaction, split and table deletion respectively. The last two sections describe how data files in the "*.deleted*" directory and old log files in archived directory should be cleaned up if they are not used any more.

6.1 Compaction

A compaction happens when there are too many store files for a region. Several files of the region are compacted as one. In current implementation old files are destroyed when the new compacted file is completely written-out to disk. But these old files could still be referred by snapshots of this table. Rather than copying these files to each snapshot which have references to them, we can simply archive these files in a common place for all snapshots, such as "*.deleted*" directory. Files are still organized as table directory. The full path for a deleted file is:

/hbase/snapshot/.deleted/<tablename>/<encoded-regionname>/<columnfamily>/<filename>

Then later operations for the snapshot would redirect to this place if it can not find the desired file following the reference path. The directory hierarchy for HBase is illustrated in Appendix B.

6.2 Split

Split only comes after compaction. A region is split into two brand-new ones. However, different from compaction, store files are not rewritten but instead creating new reference files that read off the top and bottom ranges of parent files. The old parent region is closed and offline but not deleted from HDFS. It can be still accessed from the reference file. Therefore no change is going to be made for split.

Then in MetaScanner, every region's metadata is scanned. Currently, old parent region is finally deleted if the daughters of the old region no longer keep a reference to it. However, if this parent region is still referred by other snapshots, it should not be deleted at this time but moved to the "*.deleted*" directory as what compaction does. The metadata for this region should not be removed from .META. either.

6.3 Table delete

Normally, delete a table from HBase will remove all the files from HDFS as well as remove all the table regions from .META. If there is a snapshot for this table, things are a little different.

We should go through all the metadata for the regions of this table and check the values in "*snapshot*" family. If there are values in this family, that is there are files referred by snapshots, move these files to the corresponding directory under "*.deleted*". If there is no value in this family, then this region is not referred by any snapshot and it can be removed. So is the metadata in table .META. After all referred HFiles are moved to the right place, this table can be deleted. However, those regions whose "snapshot" family is not empty should not be removed from .META. but just marked as offline.

6.4 HFile Cleanup

In previous sections, data files are not deleted as before but moved to an archived directory named "*.deleted*" if they are used by a snapshot. But how to cleanup the files under this directory if they are no longer referred by any of the snapshots. A method similar to that of deletion of split parent region can be utilized. Metadata for each region is scanned one by one in MetaScanner as before. For each column in the "*snapshot*" family, if the count value is zero, it can be deleted from the directory "*.deleted*". And if this family is empty, the entire region can be deleted from the directory "*.deleted*" and the metadata for this region can be remove from .META. table as well.

6.5 Log Cleanup

Currently log files have already been archived when they are no longer used by the table. Then in **OldLogsCleaner**, an old log file will be deleted if **LogDeleteDelegate** determine that it is not used by any other features of the system. Thus to prevent logs that are used by snapshots being deleted, just add another **LogDeleteDelegate** to the chain of LogCleanerDelegates.

7 Snapshot Operations

In this chapter, we will discuss several common operations for snapshot and how snapshot will be and manipulated for these operations.

7.1 Restore

Restoring is used to roll a table to a previous state. Restoring brings a snapshot at a particular time as an active online table in HBase so that this table has the same state as the snapshot table. Restoring is the most important and fundamental usage of snapshot. Following steps should be taken to restore a snapshot:

a) For a particular snapshot, first copy "*.oldlogs*" and all regions under <*snapshotname*> directory to the table directory */hbase/*<*tablename*>. "*.snapshotinfo*" is excluded because it is the snapshot Meta information and will not be used by an online table. Since all files under each region are reference file instead of actual data files, this step can be finished quickly.
b) In the second step, for each file in "*.oldlogs*", get the log file from log dir */hbase/.logs/* or archived log dir */hbase/.oldlogs*. Then split these logs so that all data that are not flushed at the creation time of this snapshot are now assembled in an old log file under this restored directory. This step will take some time to split the logs because the log files contain logs for several tables and could be large. After the logs are split, the "*.oldlogs*" file can be removed from this directory.
c) When all files are in place, update the .META. table with the "*.regioninfo*" in each region's directory and set the regions as online.
d) The table regions will be assigned to region servers in next Meta scanning.

At this time, the restored table is online and all HFiles under its regions are reference files instead of actual data files. Just as we do for half-reference file after split, a reference file after restoring from split will be un-referenced when compaction happens.
(Questions:

1. What if the table with the same name is still online when we want to restore a snapshot? There will be a name collision in both HDFS and .META.
2. Shall we allow rename the snapshot as a new table name? For example the snapshot is created for table "table1", can we restore the snapshot as "table2"?)

7.2 List

There would be a lot of snapshots or even several snapshots for a single table under directory "*.snapshot*". To choose one snapshot, we might need list all the snapshots information so that we can decide which one to use. For each snapshot there is a Meta info file "*.snapshotinfo*" under its directory. "*.snapshotinfo*" contains the basic information about the snapshot, such as table name of the snapshot and creation time. Therefore, to list the snapshots information, just iterate over the snapshot directory "*.snapshot*" and print the "*.snapshotinfo*" for each snapshot.

7.3 Delete

If a snapshot for a table is deleted, we should go through all its regions. For each reference file of a region, decrease the corresponding column value in .META. table by one. After all have been updated, then files can be deleted from the directory of this snapshot:

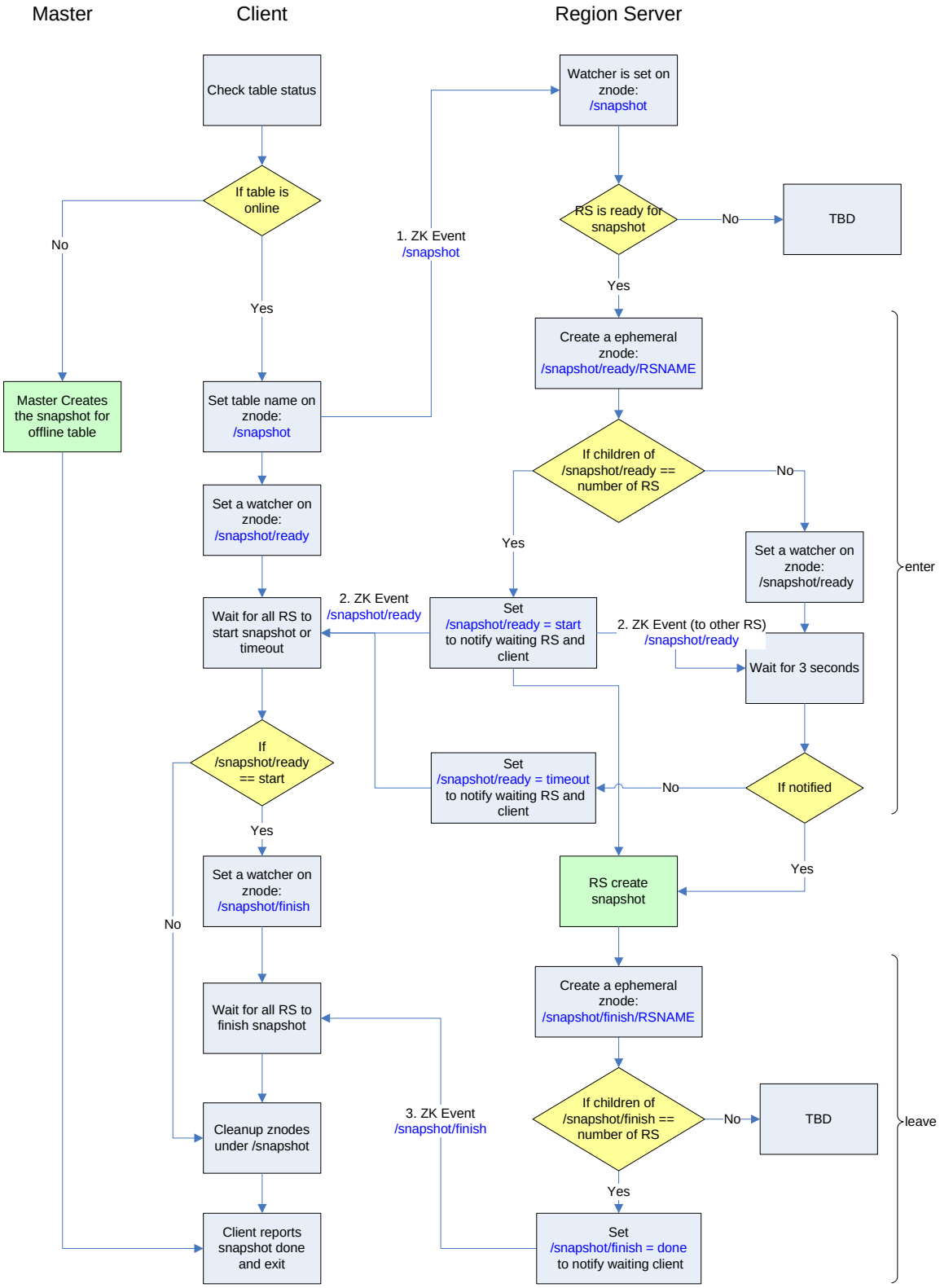
/hbase/snapshot /<*snapshotname*>

7.4 Export and Import

Sometimes, snapshot of a table would be exported to or imported from other data centers. Export of snapshot should follow the same output format as how current table is exported so that the exported snapshot can be treated the same as an exported table when it is imported with existing import facility. We can also use a map reduce job to export the snapshot. But different from exporting of a table, log files should be first split before the export is started. Then instead of read from the table API, each mapper goes to the data files (HFiles and logs) according to the references in the snapshot and reads directly from the raw data. This will make it faster than reading from API.

Import works the same as what it does right now. Because the exported snapshot has the same file format as exported table, we can use the same import class to import an exported snapshot.

Appendix A – Snapshot Work Flow



Appendix B – HBase Directory Hierarchy