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

Following are some requirements for snapshot of HBase:

1) Only the HBase admin can create a snapshot.
2) 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.
3) 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.
4) 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.
5) If one snapshot of a table is ongoing, other snapshot requests for the same table should not be permitted.
6) There should be a mechanism to export a snapshot as well as import from an exported snapshot.
7) There should be a mechanism to delete a snapshot.
8) A table could have multiple snapshots at different time. Deletion of one snapshot or even the table itself should not have influences on any other snapshots of the table.
9) There should be mechanism to list all the snapshots of a given table, including snapshot name and creation time.

3 Design Overview

Snapshot of a table can be divided into three parts:

Message Passing: Tables of HBase can be disabled and enabled. For tables that are disabled, it is not served by any region server. So we don't have to pass any message to the region servers. But for tables that are enabled, snapshot request must be first spread

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

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/client via ZooKeeper. Once these conditions are guaranteed, snapshot can be started on region servers which hold the table regions.

Snapshot Creation: Each active table in HBase is comprised of two types of data: data that have been stored on the disk and data still in the memstore. Both should be part of the snapshot.

The amount of data can be very huge in 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 to create the snapshot instead of creating actual data files. It is much faster to create references than make a copy of data. At the same time, metadata should also be included.

For the data still in the memstore, because cache flushing is time consuming, to make snapshot window narrow, we just roll the region server logger and copy the old logs. Then data in the memstore can be reconstructed from the old logs.

Snapshot Maintenance: In the above step, snapshot is comprised of references instead of actual data files, thus those data files that are referred by snapshots should not be deleted 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 moved 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.

The implementation details are described in the following three chapters.

4 Message Passing via ZooKeeper

Although clients communicate with region servers 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. The client who wants to create a snapshot will first check the table status. If the specified table is in good status and all regions of the table are online (old region after split will not be included), 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.

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 there is nothing to do for the master, the client will orchestrate the whole process. Detailed work flow is depicted in Fig. 1 and the ZooKeeper znodes hierarchy is illustrated in Fig. 2.

All the znodes created under "ready" and "finish" should be *EPHEMERAL*. That is znode will be deleted upon region server's disconnect. If a region server dies during the

snapshot, znodes created by this region server will be deleted from ZooKeeper automatically and the client will notice this and then abort the snapshot.

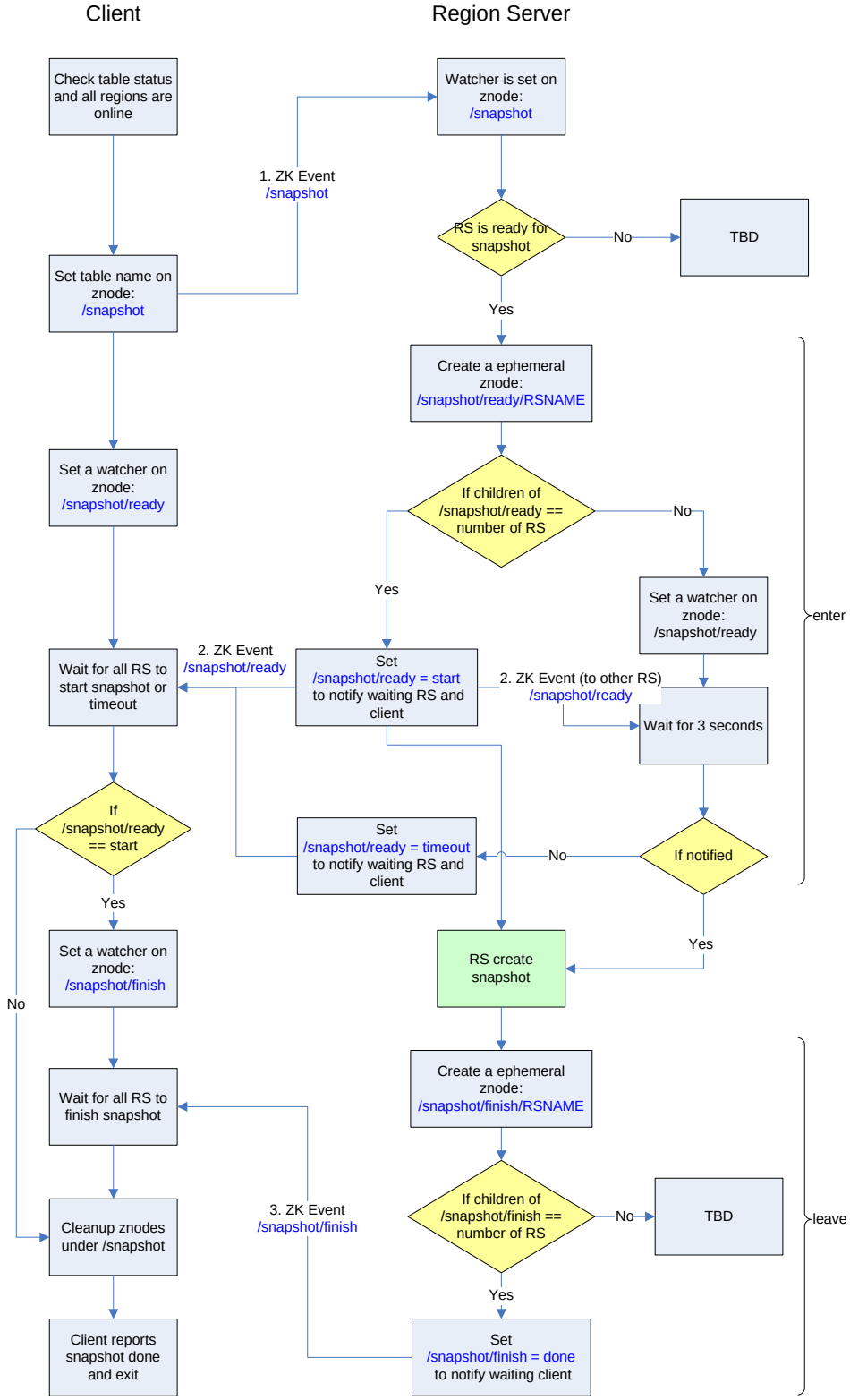

Figure 1 Snapshot Work Flow

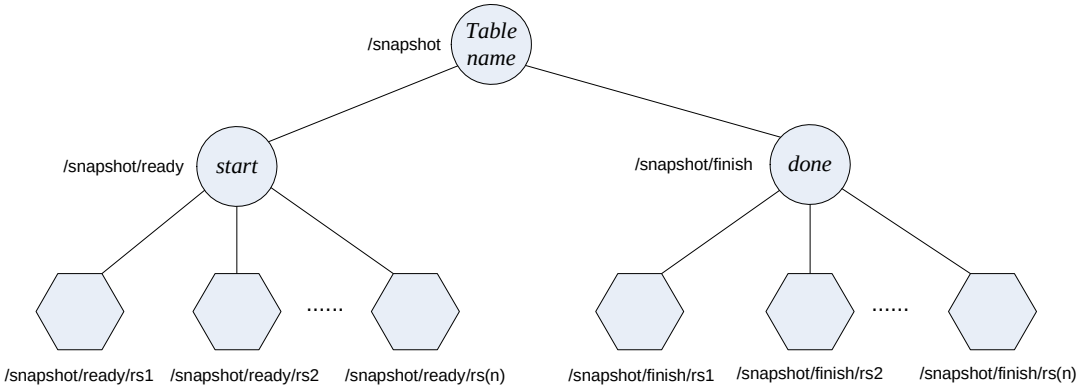

Figure 2 Snapshot Znodes Hierarchy

5 Snapshot Creation

The current state of a table is comprised of the memstore content of all currently runnigng 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. The following sections describe the handling of these data separately.

5.1 Metadata

Metadata of an HBase table is comprised of metadata of all regions. For each region of the table, there is a HRegionInfo associate with it, which contains all the meta information for this region, including table descriptor, region name as well as start key and end key. This HRegionInfo is further kept up in .META. table.

The first thought to back up the metadata is quite simple. Because this table region must be online, dumping the HRegionInfo of the region to a file ".regioninfo" under the snapshot directory of this region will obtain the metadata. The full path of the backed up metadata is:

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

An alternative method to back up the metadata is to get the metadata from .META. table first and then save the serialized metadata to the same file as above. Since the metadata for a region is only of KB, both these methods can be finished very fast.

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, 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. Then the old log files are copied to a directory named ".oldlogs" under the snapshot directory of the table. The full path of the old logs directory 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.

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 an *online* region (so old region after split is excluded), 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. 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 "reference" is created for .META. table. The qualifier is the HFile name and the value is the number of reference file that has a reference to 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.

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

Therefore, for tables that are enabled, snapshot can be created following the above three steps. But if a table is disabled, all its regions are offline and not served by any of the region servers. The whole table content is just under the directory */hbase/<tablename>* in HDFS. So we don't have to deal with the data in the memstore (Section 5.2) and snapshot can be created just by the master.

6 Snapshot Maintenance

In the above chapters, snapshot is finally comprised of log files, metadata 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 data files as well. Therefore, special care should be taken if the HFile has some references.

Following three sections discuss the snapshot maintenance for compaction, split and table deletion respectively. The last section describes how data files in the ".deleted" 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 move these file to a common place for all snapshots, such as ".deleted" directory. Files are still organized as table under this directory. The directory hierarchy for HBase is illustrated as Fig. 3. 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 reference file of the snapshot should also be updated accordingly.

(Not sure whether there will be a name collision under this ".deleted" directory)

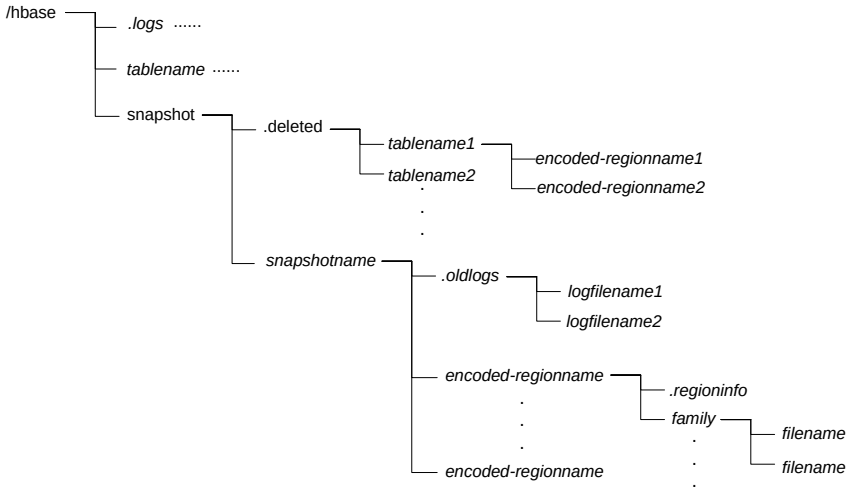

Figure 3 HBase Directory Hierarchies

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 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 split region is finally deleted if the daughters of the old region no longer keep a reference to it. However, if this old 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 old region should not be removed 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 "reference" 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 files are moved to the right place, this table can be deleted. However, those regions whose "reference" family is not empty should not be removed from .META. but just marked as offline.

6.4 File Cleanup

In previous sections, data files are not deleted as before but moved to a shadow 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 old split region can be utilized. Metadata for each region is scanned one by one in MetaScanner as before. For each file in the "reference" 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.

7 Snapshot Operations

Several operations should be provided to use the snapshots created in Chapter 5.

7.1 List

To list the snapshots for a given table, snapshot name and creation time must be included for the later operations.

(Not decides where to keep all the snapshots information, in a meta file under snapshot directory?)

7.2 Delete

If a snapshot of 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. Then files can be deleted from the directory of this snapshot.

/hbase/snapshot /<snapshotname>

7.3 Export

Currently table implementation uses a map reduce job to export an HBase table. The mapper reads data from the table and then writes to the output file. To export a snapshot of the table, mapper can go to the actual data files following the references in the snapshot. Then read directly from the raw file instead of table API.

7.4 Import

We can follow the current method to import an HBase table, where rows are read from the imported file and then inserted into the table by API. But a faster method would copy the files to the right place and then add the metadata into the .META. table. Then the table regions will be assigned to a region server in next meta scanning.