Backup and Restore

Offline Backup/Restore of Trafodion

Trafodion offline backup and restore operation are performed using HBase snapshot feature. Snapshots are a light weight way to keep the current state of a table without copying the data. After making changes to a table or in case of recovery, restoring the snapshot gives the previous state of the table. More on snapshots can be found at: http://blog.cloudera.com/blog/2013/03/introduction-to-apache-hbase-snapshots/.

Snapshots are taken for all the Trafodion tables including the Trafodion metadata tables and then exported to a HDFS location. When we need to restore the backed files from the HDFS location the snapshots are imported to the target system and then restored. The process is as follows:

Full Offline Backup

- Trafodion needs to be completely down before starting offline backup
- Snapshots are taken for all the Trafodion tables using the HBase shell snapshot command.
- Trafodion is restarted. Regular activity can resume on these tables, but they will not be included in the backup.
- Snapshots are exported one by one to HDFS folder using the TrafExportSnapshot class. For small tables this uses HDFS file copy while for large tables MapReduce is to copy HFiles. The number of mappers can vary by table. It is determined by table size and number of files to be copied for that table,
- If necessary, backup files can be archived or moved to a different cluster using hadoop distcp or other methods.

Full Offline Restore

- Trafodion needs to be completely down before starting full offline restore
- There must be no Trafodion related tables in HBase
- Snapshots that were saved in HDFS are imported one by one to HBase using the TrafExportSnapshot class. This is reverse of the copy that was done during backup.
- Once the snapshots are imported, they are restored using 'restore_snapshot' HBase shell command.

Trafodion Tables

Trafodion tables are created in HBase with names that have a format like 'TRAFODION.<schema-name>.<table-name>'. To back up all Trafodion tables, the backup tool selects and backs up all table that have a name pattern like TRAFODION.<any-schema-name>.<any-table-name>. Besides user tables, metadata tables (TRAFODION._MD_.<table-name>) and repository tables (TRAFODION._REPOS_.<table-name>) and transaction related tables (TRAFODION._MD_.<table-name>) are also backed up.

Sudo Access

The backup and restore tools can be used in both the development and clustered environment with either Cloudera or Horton Works distributions. To run the backup and restore scripts in a clustered environment, the user running the scripts needs to have sudo access to be able to run commands as HBase user (user under which HBase server runs), HDFS user and Trafodion user without requiring to enter a password. However sudo access is not needed when the scripts are run in the development environment. In a clustered environment one possibility is that root or some admin type id is used to run backup and restore. An alternative is that trafodion user is given access to run any command under the HBase or HDFS user id with a password prompt. If this acceptable from a security perspective it can be achieved by adding this line to /etc/sudoers

```
trafodion  ALL=(hbase) NOPASSWD: ALL, (hdfs) NOPASSWD: ALL, (trafodion) NOPASSWD: ALL
```

Backup Files Security

Both backup and restore use the TrafExportSnapshot which requires read and write access to these locations:

- The HBase root directory which is defined by the hbase.rootdir property. This folder is owned by HBase user
- The backup folder which needs to be either owned by the HBase user or HBase user should have read/write access to it. the Backup script tries to create the destination HDFS folder if it does not exist.

Scripts

Backup script: run_full_trafodion_backup.sh

The ./run_full_trafodion_backup.sh script performs full offline backup of all trafodion tables and copies the backup files to an HDFS location.

The command to use the script is as follows:

```
./run_full_trafodion_backup.sh -b backup_folder -u trafodion_user -h HBase_user -d hdfs_user -m mappers -l 10 -n -
o

Where:

-b backup_folder
     (Optional) HDFS path where all the Trafodion object are exported and saved
     The HDFS path needs to have a format like hdfs://<host>:<port>/<folder>/...
     If the path is not provided the script generates a path with a format like
     hdfs://<host>:<port>/trafodion-backlups/backup_<timestamp> and unless -n
     is specified the user is asked whether to confirm the use of the generated
     path.
-u trafodion user
     (Optional) The user under which Trafodion server runs. If not provided and
     if -n option is not specified the user is asked whether the default
     trafodion user 'trafodion' can be used or not. If the answer is yes then the
      default trafodion user is used otherwise the script exits.
-h hbase user
     (Optional) The user under which HBase server runs. If not provided the script
     tries to compute it and if it does not succeed it considers using the default
     HBase user 'hbase'. Unless the -n option is specified, the user is asked to
     confirm the selection afterwards.
-d hdfs user
     (Optional) The user under which HDFS server runs. If not provided the script
     tries to compute it and if it does not succeed it considers using the default
     HDFS user 'hdfs'. Unless the -n option is specified, the user is asked to
     confirm the selection afterwards.''

-m mappers
     (Optional) Number of mappers. If unspecified or 0, each snapshot will use a number suitable for its size.
-n

     (Optional) Non interactive mode. With this option the script does not prompt
     the user to confirm the use of computed or default values when a parameter
     like trafodion user, hbase user, hdfs user or backup path is not provided.
-o
     (Optional) offline. With this option trafodion will not be restarted after
     snapshots are taken.
-l
     (Optional) Snapshot size limit in MB above which map reduce is used for copy. Snapshots with size below this
value
      will be copied using HDFS FileUtil.copy. Default value is 100 MB. FileUtil.copy is invoked through a class
provided
      by Trafodion. Use 0 for this option to use HBase' ExportSnaphot class instead.



Example: ./run_full_trafodion_backup.sh  -b hdfs://<host>:<port>/<hdfs-path>  -n
```

Checks performed before backing up the Trafodion tables

The backup script performs the following checks before starting the actual backup:

- Make sure Trafodion is completely down.
- Make sure the backup folder exists, is empty and HBase user has permissions to write to it. If it does not exists it tries to create it.
- Make sure HBase snapshots are enabled.
- Make sure the user running the scripts have sudo access.
- Create the HBase home directory if it does not exist.
- Try to figure out the HBase and HDFS users if they are not provided.

Restore: run_full_trafodion_backup.sh.

The ./run_full_trafodion_restore.sh script performs full offline restore of all trafodion tables from a HDFS location

The command to use the script is as follows:

```
./run_full_trafodion_restore.sh -b backup_folder -b backup_dir -u trafodion_user -h hbase_user  -m mappers -l 10 -
n
```

```
Where:

-b  backup_folder
     (Not Optional) HDFS path where all the Trafodion object are exported and
      saved The HDFS path needs to have a format like hdfs://<host>:<port>/<folder>/...
-u trafodion user
     (Optional) The user under which Trafodion server runs. If not provided and
     if -n option is not specified the user is asked whether the default
     trafodion user 'trafodion' can be used or not. If the answer is yes then the
      default trafodion user is used otherwise the script exits.
-h hbase user
     (Optional) The user under which HBase server runs. If not provided the script
     tries to compute it and if it does not succeed it considers using the default
     hbase user 'hbase'. Unless the -n option is specified, the user is asked to
      confirm the selection afterwards.
-d hdfs user
     (Optional) The user under which HDFS server runs. If not provided the script
     tries to compute it and if it does not succeed it considers using the default
     HDFS user 'hdfs'. Unless the -n option is specified, the user is asked to
      confirm the selection afterwards.

-m mappers
     (Optional) Number of mappers. If unspecified or 0, each snapshot will use a number suitable for its size.

-l
     (Optional) Snapshot size limit in MB above which map reduce is used for copy. Snapshots with size below this
value
      will be copied using HDFS FileUtil.copy. Default value is 100 MB. FileUtil.copy is invoked through a class
provided
      by Trafodion. Use 0 for this option to use HBase' ExportSnaphot class instead.

-n
     (Optional) Non interactive mode. With this option the script does not prompt
     the user to confirm the use of computed or default values when a parameter
      like trafodion user, HBase user or hdfs user is not provided.

Example: ./run_full_trafodion_restore.sh  -b hdfs://<host>:<port>/<hdfs-path>  -n
```

Checks performed prior to starting restore

The restore scripts performs the following checks before starting the actual restore:

- Make sure Trafodion is completely down.
- Make sure the backup folder exists.
- Create the HBase home directory if it does not exist.
- Make sure HBase snapshots are enabled.
- Try to figure out the HBase, HDFS users if they are not provided.
- Figure out the HBase root directory.

Helper scripts

The backup and restore scripts use a set of shared scripts and functions that are described below:

- backup_restore_functions.sh : as set of helper functions

Examples

Backup/Restore on a cluster

Backup

To run the backup on cluster with Cloudera or Hortonworks. we can run:

```
./run_full_trafodion_backup.sh -b hdfs://<host>:8020/bulkload/backup -n

Where hdfs://<host>:8020/bulkload/backup is the hdfs location where we want the backed up snapshot to reside in
this example.
```

Restore

Restore was tested with Cloudera and Horton Works distributions. We can use something like:

```
./run_full_trafodion_restore.sh  -b /bulkload/backup -n
```

Backup/Restore on development workstation

For TrafExportSnapshot to work in the development environment, we need to copy few jar files under $MY_SQROOT/sql/local_hadoop/hbase/lib directory. Otherwise we get exceptions errors when running the ExportSnapshot MapReduce job. These workaround were figured out while testing on the workstations and there could be other ways to avoid the exceptions.

In a shell window and after sourcing in sqenv.sh, we need to run below commands once prior to running backup and/or restore.

```
cd $MY_SQROOT/sql/local_hadoop

cp  ./hadoop-2.4.0/share/hadoop/yarn/*.jar ./hbase/lib

cp ./hadoop-2.4.0/share/hadoop/mapreduce/hadoop-mapreduce-client*.jar ./hbase/lib/
```

To perform a backup on the development environment we need to determine the hdfs port first (workstations don't always use standard port). We can get the port number from the core-site.xml configuration file located under $MY_SQROOT/sql/local_hadoop/hadoop/etc/hadoop/. In this example, it's 28400:

```
bash-4.1$ cat $MY_SQROOT/sql/local_hadoop/hadoop/etc/hadoop/core-site.xml

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

     <property>

         <name>fs.default.name</name>

         <value>hdfs://localhost:28400</value>

     </property>
```

Backup

To run the backup using the port number from above, the command is:

```
./run_full_trafodion_backup.sh -b hdfs://localhost:<port-number>/bulkload/backup -n
```

Restore

To run the restore:

```
run_full_trafodion_restore.sh -b hdfs://localhost:<port-number>/bulkload/backup -n
```

Testing

The test cases that were performed so far include:

Backup/Restore on one system with HBase 0.98

All Trafodion tables created in HBase 0.98 were backed up and exported to a hdfs location. Then all the Trafodion tables were dropped. Then the backup files were restored them from the backup location to the same HBase .98. This type of tests was done on:

- Workstation with Trafodion 0.9 and HBase 0.98 with a Cloudera distribution (development environment). Test succeeded.
- Cluster with 4 nodes that has Trafodion 0.9 and HBase 0.98 (HortonWorks distribution). Test succeeded.
- Cluster with 2 nodes that has Trafodion 0.9 and HBase 0.98 (Cloudera distribution). Test succeeded.

Backup/Restore using two different systems with Trafodion 0.9 and HBase 0.98

The steps involved in these tests are:

- Backup of the source Trafodion to a HDFS location
- Moving the hdfs folder to another system with HBase .98. This step involved copying the backup files from HDFS to the local disk then using scp linux command to move the files to the target system, then copying the backup files to a HDFS location on the target system. Another way to move the backed up files is to use hadoop distcp but currently it does not work because the IP ports associated with distcp are not open.
- Restoring the backup on the target system from the backup files

This types of tests was done:

- From a cluster with 4 nodes and Horton works distribution to a cluster with 2 nodes and Cloudera distribution. Tests succeeded
- From a workstation to another workstation (development environment) having both the HBase from Cloudera distribution. Tests succeeded

Backup/Restore using two different systems both having Trafodion 0.9 where the source system is HBase 0.94.6 and the target is HBase 0.98.

While working on this test the main issues we faced are summarized below:

- Exporting Snapshot that are empty (table is empty) from he HBase.94.6 fails and generates exception. This is a known issue which is described in https://issues.apache.org/jira/browse/HBASE-8199 . This export issue was fixed in HBase 0.94.7 but not in HBase0.94.6.
- There is compatibility issues between HBase in general (including snapshots) between HBase .94.6 and later version including the 0.98. . Because of this issue the import of the HBase .94.6 snapshot to HBase .98 does not work
- One way to solve this compatibility issue may be to upgrade from HBase 0.94.6 to 0.98.x before doing the backup if we want to restore to a 0.98 HBase. The upgade it self from HBase 0.94.6 to HBase 0.98 needs to happene in two steps: upgrade from HBase 0.94.6 to HBase 0.96. and then upgrade from HBase 0.96 to HBase 0.98. the upgrade steps are described here http://HBase.apache.org/book/ch03s04.html
- This test case did not succeed because of the issues/limitations listed above.