HBASE-18659 Use HDFS ACL to give user the ability to read snapshot directly on HDFS

Link of this document:
https://docs.google.com/document/d/1D2iAdbrW5CcKc2SthJBXA1n2tTMTftuVaFtxbOWFuqM

Background

In HBase, scan is expensive. Fortunately, HBase provides TableSnapshotScanner to directly scan hfiles, which consumes less resource. But only super users have read permission over hfiles, we solve the problem by giving users granted in HBase the read permission over hfiles.

Basic idea

Use HDFS ACLs to set users read permission over hfiles.
HDFS ACL supports an "access ACL", which defines the rules to enforce during permission checks, and a "default ACL", which defines the ACL entries that new child files or sub-directories receive automatically during creation.
Learn more details about HDFS permission:
https://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-hdfs/HdfsPermissionsGuide.html

The hfiles of HBase are origanized in the following ways:

- tmp data directories

/{hbase-rootdir}/.tmp/data
/{hbase-rootdir}/.tmp/data/{namespace}
/{hbase-rootdir}/.tmp/data/{namespace}/{table}

- data directories

/{hbase-rootdir}/data
/{hbase-rootdir}/data/{namespace}
/{hbase-rootdir}/data/{namespace}/{table}

- archive data directories

/{hbase-rootdir}/archive/data
/{hbase-rootdir}/archive/data/{namespace}
/{hbase-rootdir}/archive/data/{namespace}/{table}

- snapshot directories

/{hbase-rootdir}/.hbase-snapshot
/{hbase-rootdir}/.hbase-snapshot/{snapshot-name}

Generally, table directories are firstly created in [tmp data directories] and then move to [data directories] if create table successfully.

The hfiles of table [data directories] are archived to [archive data directories] when delete table/ delete CF / merge / compact … happens.
A snapshot will reference to hfiles under these data directories.
If user scan a snapshot, then he must own permissions to access [snapshot directories], table [data directories], table [archive data directories], table [tmp data directories].

Based on this, the idea is very simple: add HDFS ACLs in hfiles for users with HBase read permission.
HBase provides different kinds of permission:

- GlobalPermission

  Add access and default HDFS ACLs in the parent of namespace directories(named **global** directories in the following doc):

      /{hbase-rootdir}/.tmp/data
      /{hbase-rootdir}/data
      /{hbase-rootdir}/archive/data
      /{hbase-rootdir}/.hbase-snapshot

- NamespacePermission

  Add access and default HDFS ACLs in **namespace** directories:

      /{hbase-rootdir}/.tmp/data/{namespace}
      /{hbase-rootdir}/data/{namespace}
      /{hbase-rootdir}/archive/data/{namespace}
      /{hbase-rootdir}/.hbase-snapshot/{snapshot-of-namespace-tables}

- TablePermission

  Add access and default HDFS ACLs in **table** directories:

      /{hbase-rootdir}/.tmp/data/{namespace}/{table}
      /{hbase-rootdir}/data/{namespace}/{table}
      /{hbase-rootdir}/archive/data/{namespace}/{table}
      /{hbase-rootdir}/.hbase-snapshot/{snapshot-of-table}

  Add access HDFS ACLs in **namespace** directories:

      /{hbase-rootdir}/.tmp/data/{namespace}
      /{hbase-rootdir}/data/{namespace}
      /{hbase-rootdir}/archive/data/{namespace}

- Table CF or CQ Permission:

  Ignore because can’t take a snapshot for one CF or CQ of a table

Also need to set other permission to --X for following directories (It means the permission of there directories is 751 or 701):
global directories:

    /{hbase-rootdir}/.tmp/data
    /{hbase-rootdir}/data
    /{hbase-rootdir}/archive/data
    /{hbase-rootdir}/.hbase-snapshot

parent of global directories:

    /{hbase-rootdir}/.tmp
    /{hbase-rootdir}/archive
    /{hbase-rootdir}

/{parent-of-hbase-rootdir}
/

In this way, a user can finally access table data directories.
Note:
There may be a confusing problem that why set other permission to --X instead of adding user HDFS ACLs in these directories?
Because HDFS has a config which limits the max ACL entries num for one directory or file:
dfs.namenode.acls.max.entries = 32(default value)
If add HDFS ACLs for these top level directories, the number of ACLs must exceed 32.

As the following picture shows, if HDFS ACLs feature is enabled, the previous 8 ACL entries are set for all directories by default, so we can add up to 24 ACL entries(maybe 12 HBase users if each user has a default and an access ACLs).
And this is a limitation to this feature, if grant too many global, namespace, table READ users, the feature will not work right.

	# file: /hbase/c3tst-cluster/data/ns1 # owner: hbase_tst # group: supergroup
1	user::rwx
2	default:user::rwx
3	group::r-x
4	default:group::r-x
5	mask::r-x
6	default:mask::r-x
7	other::--x
8	default:other::--x
9	user:ns_user:r-x
10	default:user:ns_user:r-x

In order to make this feature a plugin, we add a master coprocessor to implement it.

Implementation

There are many details when implementing this feature because many master operations will modify or move table directories.

master operation	coprocessor hook
start master	(Skip delete /{hbase-rootdir}/.tmp directory because ACLs are set at these directories;) Set global directories permission to 751;
create namespace	Create namespace directories if not exists used for default ACLs inherited;
delete namespace	Because namespace archive directories may still exists: 1. Delete namespace archive HDFS ACLs; 2. Delete empty namespace directories used for default ACL inherited;
snapshot	Add HDFS ACLs for users with global, namespace, table read permission for /{hbase-rootdir}/.hbase-snapshot/{snapshot-name}
create table	1. Create table directories if not exists used for default ACLs inherited; 2. Add table HDFS ACLs for owner;
truncate table	Because table directories are recreated: 1. Create table directories if not exists used for default ACLs inherited; 2. Add table HDFS ACLs for users with table read permission
delete table	Because table directories are archived and table user acls are deleted: 1. Revome ACLs from table archive directories; 2. Remove namespace HDFS access ACLs for users with table read permission
modify table	* If enable this feature for table: 1. Create table directories if not exists used for default ACLs inherited; 2. Add HDFS ACLs for users with global, namespace, table read permission; * If disable this feature for table: 1. Delete empty table directories used for default ACL inherited; 2. Remove table HDFS ACLs for users with global, namespace, table read permission; 3. Remove namespace access ACL for users only own this table read permission;
grant	Judge this grant adds read permission or removes read permission, if delete, do revoke logic. * Grant global read permission Add global HDFS ACLs * Grant namespace read permission Add namespace HDFS ACLs * Grant table read permission Add table HDFS ACLs

revoke	* Revoke global read permission Remove global HDFS ACLs and skip namespace or table directories which have the user read permission * Revoke namespace read permission Remove namespace HDFS ACLs and skip table directories which have the user read permission * Revoke table read permission Remove table HDFS ACLs; Remove namespace access HDFS ACLs;

Meanwhile, in order to skip some duplicate HDFS ACLs modification, we add a ‘m’ CF in acl table and mark if the user HDFS ACLs is set.
For example, grant user ‘u1’ with table ‘ns1:t1’ read permission and grant ‘u1’ with namespace ‘ns1’ read permission, then skip set ‘ns1:t1’ table directories HDFS ACLs for namespace user ‘u1’.

We create empty directories when create namespaces or tables to add ACLs for these directories:

    /{hbase-rootdir}/archive/data/{namespace}
    /{hbase-rootdir}/archive/data/{namespace}/{table}

However, HFileCleaner will delete these empty directories and ACLs are lost, so we add a new cleaner plugin named SnapshotScannerHDFSAclCleaner to keep directories used for this feature can be reserved.

Configuration

1. Firstly, make sure that HDFS ACLs is enabled and umask is set to 027:
   dfs.namenode.acls.enabled = true
   fs.permissions.umask-mode = 027

2. Add master coprocessor:
   hbase.coprocessor.master.classes =
“org.apache.hadoop.hbase.security.access.AccessController
,org.apache.hadoop.hbase.security.access.SnapshotScannerHDFSAclController”

3. Enable this feature:
   hbase.acl.sync.to.hdfs.enable=true

4. Modify table scheme to enable this feature:
alter 't1', CONFIGURATION => {'hbase.acl.sync.to.hdfs.enable' => 'true'}

Limitation

1. If we enable this feature, some master operations described in the implementation for tables which enabled this feature will be slower as we need to set ACLs to all the existing files.
2. HDFS ACLs supports up to 32 ACL entries which includes four fixed user : owner, group, other and mask and each has 2 ACL entries (access and default). This means this feature can only support up to 12 users to a table besides users granted to table namespace and global.
3. There are other cases that this coprocessor has not handled or could not handle. Such as a reference link to another hfile of other tables.