FAST LTA

Long Term Archiving

Silent Cube Background Paper

Versioning (Release Build 1.00.18.4604)

Adam Burg, Jörg Jünger

21. Januar 2010

Versioning

Terms

Description

Detailed example (Linux client)

Versioning

Terms

commit
Operation that invokes the creation of a new version of the file in the file system. Essentially makes the current data in the local cache persistent. Usually the commit is triggered by setting the read-only flag on (i.e. removing the write bits from) a file.

commit phase
The period between the commit of a file and its becoming safe. Typically, during the commit phase, a file has the attribute `R` set in Windows, and also attribute `A`.

top-revision

The most recent revision of the file. In technical terms, it is the revision of the inode that is stored directly in the `Inode` table.

non-top-revision
A previous revision of the file that has been overwritten by new commits. In technical terms, all revisions which are stored in the `InodeVersion` table are *non-top-revisions*.

active revision
A revision of the file that has been activated for access. If the active revision is not the top revision, the file size information in file listings and the hash information in `safe`-files corresponds to the active revision. In other words, a file listing will always show the file size of the active revision, and not necessarily of the top revision.

Description ¶

The versioning volumes allow to virtually overwrite files which are already safe. In doing so, multiple revisions of the file are created, which can be accessed through special methods.

The versioning volume types introduce a new kind of special file in addition to the `safe`-files: the version-file. This file contains a list of the previous revisions of the actual file it corresponds to. A file that has only the initial version doesn't have have an associated `version`-file. This means that as long as no files are overwritten, a volume has only the data files and `safe`-files. The `safe`-file mechanism works as usual: whenever the file becomes safe, the `safe`-file appears (this has one special implication on versioned volumes, though, which will be addressed below).

Whenever an old revision is activated and opened, the corresponding file is downloaded to the `data` directory. If the top revision is active and the file is modified to create a new revision, all modifications are executed on the file in the `data` directory. This is the reason why, as you will see later on in this description, you always have to make sure that there is no file in the `data` directory before changing revisions.
Keep in mind that the access timestamp set for a file is valid for *all* revisions of it, i.e. the retention time is not versioned.

Create initial revision ¶

The following is a description of the steps to take to create the initial version of the file:

1. Create a file.
2. Write some data to it.
3. When you are done, close the file.
4. Set the read-only flag (or remove the write bits).
5. Wait till the safe file appears.

The file is in commit phase between steps 4 and 5. During this phase, it can't be changed back to writable state, it can't be truncated, opened for writing, etc. However, it can be deleted (unlink-ed). This is a special case, later on, once the file has revisions, it can't be deleted anymore in the commit phase.
At this point, the file has no corresponding version-file, because there is only one revision: the top one.

Create a new revision ¶

Here is how new revisions can be created now:

1. Remove the read-only flag from the file (or set +w).
2. Open the file. In technical terms, use O_TRUNC to open the file. Many editors and applications already do this if the target file exists. Deleting the file first will **not** work.
3. Write some data to the file.
4. Close the file handle.
5. Set the read-only flag.
6. Wait till the safe file appears.

A note about step `6`: once a file is committed, its top revision becomes "unsafe" again, since it has to be ingested. This means that during the commit phase of a versioned file the safe file will `disappear`. When

it reappears the committed revision has become safe (again).
After step 5 a `version`-file will appear, which contains the initial revision. Every time a new revision is committed, the top revision at that moment is stored in the `InodeVersion` table in the database, and thus will be shown in the `version`-file.

Accessing previous revisions

Accessing previous revisions is controlled through commands written to the `version`-file. As mentioned before, when reading from this special file, the contents will list all the previous revisions of the file (excluding the top one). However, the version file can be written to, at any address (the actual contents will not be overwritten, since it is a virtual file). The written data is evaluated, and any known commands are executed. Every command must be terminated by a newline or line feed character (`\n` or `\r`); space characters at the end are trimmed.
The following commands are accepted:

switch <revision>

Switches the active revision to the specified revision.

revert

Switches the active revision back to the top revision. Any local files in the data cache are deleted to avoid confusion with other revisions.

Writing the command can be as easy as

echo "switch 0" > /var/fast/sc/exports/3529134965894154639O/mount/vid.avi.version

or

echo "revert" > /var/fast/sc/exports/3529134965894154639O/mount/vid.avi.version

Switching the revision has a few limitations:

1. Switching is not allowed if the file is in commit phase. This also implies that whenever switching is allowed, the `safe`-file exists.
2. Switching is not allowed if there is a local file in the cache. You can check in the `safe`-file (*online* attribute) whether there is a local file. It can be thrown away using the `revert` command before switching. *Be careful: any local changes will be voided*.
3. If the target revision is already the current active revision, the `switch` command will be ignored, however, there will be no error.

4. Switching will fail if the currently active revision is neither the top revision (corresponds to a `revert`) nor the new revision (corresponds to `#3`). In other words: you have to alternate between `switch` and `revert`, you can't `switch` directly between old revisions.
5. The target revision must be available.

Here is how you would go about accessing a previous revision. To retrieve the revision's number you can check the entries in the `version`-file.

1. Make sure there are no local modifications. Check the `safe`-file's *online* attribute or just send a `revert` command if you are sure about it.
2. Switch to a previous revision using the `switch` command.
3. Open the file for **reading only** (opening in write mode would fail).
4. Read from it.
5. Close the file.
6. Revert back to the top revision using `revert`.

If possible and available, the previous revision will be downloaded using random access.

Deleting files from versioned volumes

Deleting files from versioned volumes is, like with every other volume, not possible, with a few exceptions:

- The initial revision of a file can be deleted as long as it is not ingested (this is conformant with type 1 and 2 volumes).
- On retentioned volumes, the retention period has passed.

On versioned volumes the default retention time of a volume is only set when the initial revision of a file is created. Subsequent overwrites will not extend the retention time (however, the client may choose to do so manually). When a versioned file is deleted after retention, all its associated versions are deleted with it.

Detailed example (Linux client)

Create a file:

dd if=/dev/urandom of=versiondemo count=1 bs=1M

1+0 records in

1+0 records out

1048576 bytes (1.0 MB) copied, 0.193661 s, 5.4 MB/s

Here is how it looks now:

ls -la versiondemo

-rw-r--r-- 1 abg abg 1048576 2009-08-26 14:20 versiondemo

Commit it:

chmod -w versiondemo

Wait till it becomes safe (over NFS mounts it's better to check it directly instead of listing the whole directory due to the directory cache):

ls -la versiondemo.safe

-r--r--r-- 1 abg abg 1024 2009-08-26 14:22 versiondemo.safe

Check the hash. Note the *online* property: a value of *1* means there is a local file.

more versiondemo.safe

hash=04caf4521d883433285ec208633832ad074bd58db54aae7c8f7abee5c88296e65460a3c66689b7
5593ee3ce5c1ad81f2832c6bd1b42522d22d1f82bcf656a484

..

online=1

..

top_revision=0

active_revision=0

Set the file to writable again and overwrite it:

chmod +w versiondemo

dd if=/dev/urandom of=versiondemo count=1 bs=1M

1+0 records in

1+0 records out

1048576 bytes (1.0 MB) copied, 0.183621 s, 5.7 MB/s

Commit it:

chmod -w versiondemo

Check contents of the `version`-file. The first revision of the file has the revision number `0`.

more versiondemo.version

revision,hash,ingest_iso8601,original_size,dbid

0,04caf4521d883433285ec208633832ad074bd58db54aae7c8f7abee5c88296e65460a...,2009-08-26T14:20:55+02:00,1048576,45

Wait till it's safe. Notice that the hash has changed - it's the hash of the new revision.

more versiondemo.safe

hash=753a7a9249445dc47413ebadce6e66cd5b8881cd5620138c43ebc89a693c9431bafae42f89dfd93402aaefc9fe0d2ca0adcdb3ca5c2a504a2eb840ae3704fbc5

..

online=1

..

top_revision=1

active_revision=1

Notice that the `online` argument says, that there is a local file. We have to get rid of this before we can switch:

echo "revert" > versiondemo.version

more versiondemo.safe

hash=753a7a9249445dc47413ebadce6e66cd5b8881cd5620138c43ebc89a693c9431bafae42f89dfd93402aaefc9fe0d2ca0adcdb3ca5c2a504a2eb840ae3704fbc

5

lookupkey=3944c72ad214cf035c84619e5b586208aea3e2248141b25413f14aec0c94285ca57cc95d59a7de21af829a95c89ac2fce71f26b0e4c4863d5b45622fff

..

online=0

..

top_revision=1

active_revision=1

Now switch to the first revision:

echo "switch 0" > versiondemo.version

The file information shown will now belong to the activated revision. The `safe`-file will show the top and active revisions:

more versiondemo.safe

hash=04caf4521d883433285ec208633832ad074bd58db54aae7c8f7abee5c88296e65460a3c66689b75593ee3ce5c1ad81f2832c6bd1b42522d22d1f82bcf656a484

..

online=0

..

top_revision=1

active_revision=0

Check the hash:

sha512sum versiondemo

```
04caf4521d883433285ec208633832ad074bd58db54aae7c8f7abee5c88296e65460a3c66689b75593e
e3ce5c1ad81f2832c6bd1b42522d22d1f82bcf656a484  versiondemo
```

You can see that this is the same hash that was listed in the `safe`-file after the initial ingest.

Let's go back to the most recent revision and recalculate the hash:

echo "revert" > versiondemo.version

sha512sum versiondemo

```
753a7a9249445dc47413ebadce6e66cd5b8881cd5620138c43ebc89a693c9431bafae42f89dfd93402a
aefc9fe0d2ca0adcdb3ca5c2a504a2eb840ae3704fbc5  versiondemo
```