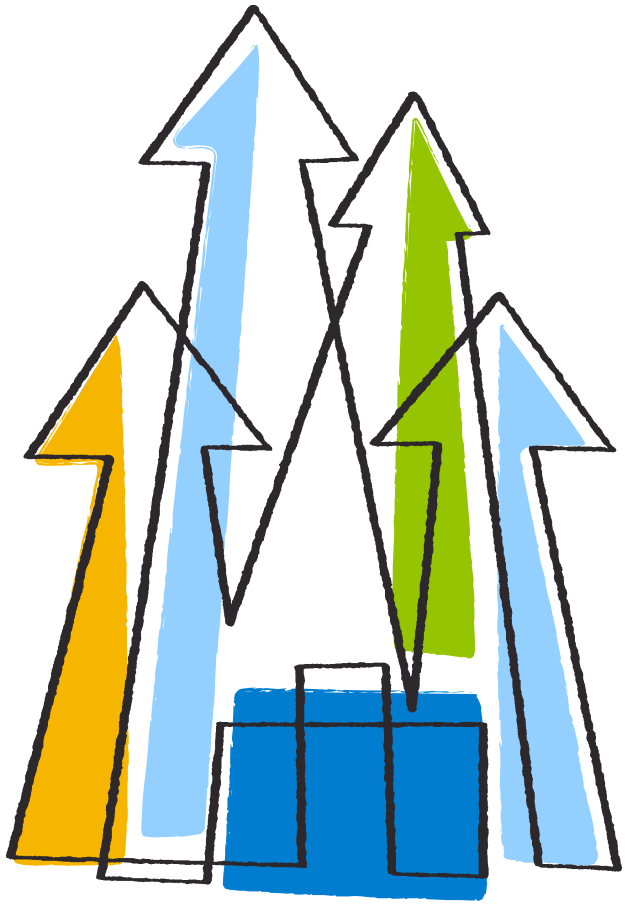
NetApp

Go further, faster®

NFS Client GSS Context Management Progress



Andy Adamson

Connectathon 2013

Two GSS Context Management Problems

- GSS Context expires with un-flushed buffered WRITEs
- Kerberos credentials destroyed (kdestroy) but GSS context is still valid
- Both solutions in the RFC stage, with working code

NFS Client GSS Context Creation & Update

- TGT is obtained via *kinit* with a lifetime TGT-L
- User accesses NFS share which triggers a TGS to be obtained with lifetime TGS-L
- TGS-L is (usually) less than TGT-L
- The TGS is used to setup a GSS context
- GSS context lifetime == TGS-L

NFS Client GSS Context Creation & Update

- Each GSS RPC call checks the GSS context lifetime against the current time
- If the GSS context has expired, an upcall is performed to renew the context
- This is only possible if the TGT is still valid. If it is, another TGS is obtained and a new GSS context is created
- If the TGT has expired, the renew upcall fails and the user has no GSS context so access to the NFS share stops

GSS Context and Buffered WRITEs

- Current code allows a GSS context to expire with un-flushed buffered WRITEs
- A new feature solves this by setting up a credential key expiry *watermark*, a “line in the sand”, watermark seconds from the end of a GSS context lifetime.
- The watermark value is based on the dirty_expire_interval (default 30 seconds) which is the longest a page can remain un-flushed in the buffer cache.

GSS Context and Buffered WRITEs

- Each GSS RPC call checks the GSS context lifetime minus the watermark against the current time
- If the GSS context lifetime is within the watermark, an upcall is performed to renew the context
- If the upcall fails, a flag is set in the RPC GSS credential associated with the GSS Context
- At the beginning of the buffered write code path, the flag is checked.
- If the flag is set, the inode is flushed and WRITEs are sent with NFS_FILE_SYNC

GSS Context and Buffered WRITEs

- Issues to resolve
  - Watermark value needs to be long enough to flush all buffered WRITEs and (possibly) send COMMITs
  - Some dependency on work load
  - Currently set to 10 seconds past dirty_expire_interval
  - May be a module parameter
- Solves issues for NFSv3 and NFSv4
- NFSv4.1 can use SP4_mach_cred to allow the machine credential to flush buffered WRITEs on GSS context expiration

GSS Context destruction upon kdestoy

- Currently kdestroy has no effect on the associated GSS context
- User can log off (kdestroy), but NFS GSS access is still active
- Problem: how to signal the Kernel GSS layer upon kdestroy
- To solve this, I chose the Kernel Keyring service

GSS Context and Kernel Keyring

- We register a new key type called *gss-ctx* in the auth_gss module
- *gss-ctx* is based on the *user* key type – we use the default functions for instantiate, match, revoke, describe and read.
- Change the destroy function
- We add two new user programs *gss_login* and *gss_logout*
  - (not attached to the names!)

gss_login

- gss_login calls kinit, and then instantiates the gss_ctx kernel key
- As an added bonus, we store the location of the Kerberos credential cache in the key which GSSD then uses.
- No more searching for the Kerberos credential cache

gss_logout

- gss_logout calls kdestroy, and then destroys the gss_ctx kernel key
- The gss_ctx key destroy function flushes all GSS context buffered I/O and then destroys the RPC GSS credential and associated GSS context
- For now, it uses the big hammer and calls sys_sync to flush all data to all file system

NetApp

Thank you