NFSv4.2 Inter SSC Prototype: intra,inter,sync,async

Olga Kornievskaia
kolga@netapp.com
Connectathon 2017
Presenter Title
Date

Existing code

- Upstream kernel contains synchronous intra SSC prototype + needed VFS changes for Linux client and server (Anna Schumaker, NetApp)
- In the works is support for inter SSC and asynchronous copy for Linux client and server (Andy Adamson, Olga Kornievskaia, NetApp)
- Upstream NFSTest includes test cases for intra and inter SSC (Jorge Mora, NetApp)

Client implementation details

- Application (NFSTest) makes a copy_file_range() system call into VFS layer
- Removes VFS restriction from cross-devices copies
- VFS calls into NFS's copy_file_range()
- NFS determines if request is for intra or inter copy (checks superblocks)
- If inter, it sends COPY_NOTIFY to source server
- Latest code will then choose to send COPY requesting an asynchronous copy (will resend it as sync if ERR_OFFLOAD_NOREQS received)

Client details (cont)

- For inter, if application cancels in-flight COPY, OFFLOAD_CANCEL is sent to the source server
  - Unable to send OFFLOAD_CANCEL to destination server (COPY might start on dst server)
- If server accepted async copy request, client will go wait on the CB_OFFLOAD
  - CB_OFFLOAD can arrive prior to COPY reply, keeping a list of pending callbacks to check before waiting on CB_OFFLOAD

Client details (cont)

- If application cancels async COPY after reply received, OFFLOAD_CANCEL is sent to source and destination servers (or server for intra)
- OFFLOAD_CANCEL is sent as an asynchronous RPC in the context of nfsiod_workqueue
- Finally, client sends COMMIT if needed
- If COPY or CB_OFFLOAD return a partial copy, result is propagated to the application

Client details (cont)

- Handling reboot of the destination server
  - Client's heartbeat should discover server reboot, it needs to wakeup waiting copies and mark them. If waiting on copy client wakes up to find copy marked dest server rebooted, it needs to start copy from scratch
- Handling reboot of the source server
  - Destination server will still send back CB_OFFLOAD and wake up waiting copy and restart the copy from the new offset
  - Client's heartbeat might discover server reboot, it will mark that copy as source server rebooted
  - Client will send a new COPY_NOTIFY and COPY

Destination server implementation details

- NFSD determines if COPY is intra or inter and if synchronous or asynchronous
- For inter, NFSD uses NFS4.1 protocol and mounts the source server (mount src_add:/ on an internal mount point). Cleans up mount when done with copy
- If COPY is asynchronous, NFSD queues up a work item for a single threaded workqueue, then generates and saves in the list a unique copy stateid and replies back to the client
  - COPY arguments available in NFSD running context must be copied for the workqueue

Dest server details (cont)

- To be able to do partial copy, copy is broken into 4MB chucks and then is done in a loop by calling into VFS copy_file_range() which then
  - for “inter” calls into do_splice() to copy via NFS4.1 READs and local file writes
  - for “intra” depending on underlying functionality will do clone or do_splice (no other FS does copy_file_range)
- Once asynchronous COPY is done, results are queued for the callback workqueue and sent via CB_OFFLOAD

Dest server details (cont)

- Inter COPY requires NFS client functionality via a new kernel mode that contains nfs42_ssc_open():
  - Creates an “open” that fills in both VFS and NFSv4 data structures
  - Does a GETATTR on the COPY SAVE_FH file handle
  - Creates an inode, a dentry, a struct file, an nfs_opencontext (with ca_src_stateid) all bypassing most of the normal interfaces
  - Places the “open file” in the directory client namespace directly under the mount point with a fake name
- And nfs42_ssc_close() function:
  - Needs to bypass normal close that triggers CLOSE on the wire

Dest server details (cont)

- Reboot recovery of the source server
  - READ would receive BAD_SESSION error triggering session and clientid recovery, as well as any state – need to mark the open stateid as something not to be recovered and fail READ (EIO or BAD_STATEID)
  - Coming from NFS to VFS we ”lose” NFS errors that are translated into generic errors
  - Server can choose to either return incomplete read with NFS4_OK or an error say NFS4ERR_EIO?

Source server implementation details

- NFSD receives COPY_NOTIFY, it creates a unique copy stateid that keeps track of its parent's open/lock stateid and adds to the global list of copy stateids
- NFSD receives READ, it checks normal list of stateids, and then also the list of copy stateids before either finding a match or returning an error

Server details (cont)

- NFSD can receive OFFLOAD_CANCEL
  - Source server removes specified stateid from the list of copy stateids
  - Destination server looks thru the list of copy stateids and marks it cancelled. Once server calls into vfs_copy_file_range() no way to interrupt it. Relies on READs to fails with BAD_STATEID to fail vfs_copy_file_range()

Questions

- Removal of cross-device check in VFS has been questioned
  - alternative an ioctl but “cp” will use copy_file_range()
- What layer is responsible for completing the request (looping until copy is done)? Each layer?
  - Server tries to loop but with reboot can return partial result
- Linux client has no API to query status of the ongoing copy (finding no use for OFFLOAD_STATUS)
  - Looping on the server for 4MB vfs_copy_file_range() allows for intermediate status (without no status is available)

Questions

- What kind of errors could be sent in CB_OFFLOAD and how to interpret them?
  - If source server reboot can trigger an error on CB_OFFLOAD (how to decide what error is recoverable and what is not)?
  - If source server rebooted before any data was read (partial copy data = 0), how should the client treat that to seamlessly handle reboot recovery of server
  - Is there a difference between CB_OFFLOAD with partial copy and NFS4_OK and NFS4ERR_x?
- IF CB_OFFLOAD returned and error and partial copy, what is assumed about “committed how” flag?

Questions

- Copy beyond end of file: EINVAL or short copy?
  - Still confused about IETF discussion
  - Server should ignore invalid arguments and return a short copy
  - Client can get 3 choices:
    - CB_OFFLOAD NFS4_OK bytes=partial copy
    - CB_OFFLOAD EINVAL bytes=partial copy
    - CB_OFFLOAD EINVAL bytes=-1
- No support for partial copy with error

Questions

- How to determine on the client if to request a synchronous copy or asynchronous one (eg., size, mount option)?
- How to determine on the server when to turn an asynchronous request into synchronous (eg., “short” copy – a configurable parameter)?
- What conditions on the server should trigger ERR_OFFLOAD_NOREQS to the async copy request (eg., “too many” queued copies?)

Thank you